RICARDO URIBE

A FREQUENCY MODULATION MUSIC SYSTEM

BY

ROBERT MICHAEL KAMINSKY

B.S., Rose-Hulman Institute of Technology, 1980

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1981

Urbana, Illinois

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to my thesis advisors, Prof. Ricardo Uribe and Prof. William Jenkins, for their guidance on this project and helpful suggestions in the writing of this manuscript.

Special thanks to Jim Graf for explaining the use of some of the lab equipment, Shaw Moldauer and Glenn Poole, for their suggestions concerning the Processor Board, and Prof. Ricardo Uribe, for his patience and helpfulness throughout the course of this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This thesis describes the FMMS (Frequency Modulation Music System) which uses frequency modulation (FM) techniques to produce sounds which have dynamic frequency spectra. By using FM, a multitude of musical timbres can be produced. The system was designed and built in the ADSL (Advanced Digital Systems Laboratory) and integrates into the laboratory's Music System. This system is controlled by an 8085 CPU (Central Processing Unit) and interfaces to an organ keyboard, an amplifier and speakers, disk memory and several different music synthesizers. Figure 1.1 shows a block diagram representation of this arrangement.

The text is organized to first acquaint the reader with some of the theory behind FM and then goes on to present the considerations which were made in the actual designing of the system. Chapter two describes the mathematical theory behind frequency modulation, using spectral

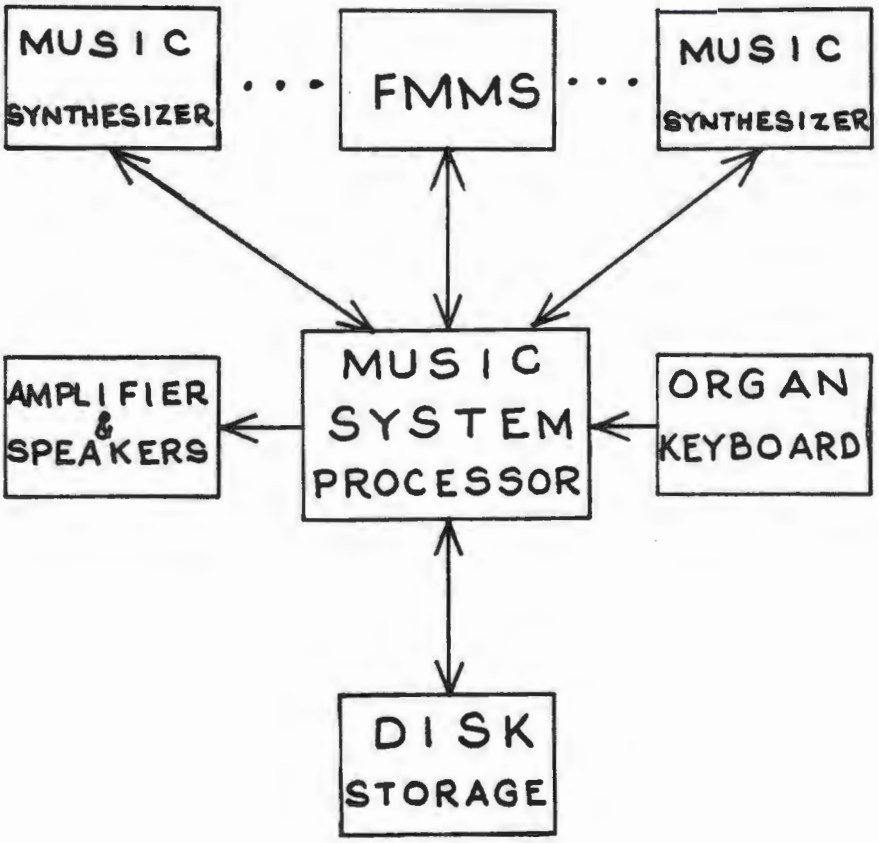


Figure 1.1 THE ADSL MUSIC SYSTEM

analysis to explain how a tone's timbre (sound character) can be enriched. Chapter three describes the hardware block structure of the FMMS, whereas chapter four goes into the details concerning the hardware design considerations. Chapter five is concerned with the system software, and in chapter six, some ideas are given on how to expand the FMMS to handle multivoice, followed by some concluding remarks.

1.1 Design Overview

The FMMS is chiefly an experimental project. The primary goal is to allow the user enough flexibility to create a variety of sounds presently unobtainable with the ADSL Music System.

The FMMS consists of four vector boards which are mounted in a rack in the ADSL Music System cabinet. The first three boards manipulate the data sent to them by the fourth board (the Processor Board). The hardware consists of pipeline registers and adders, a sine wave look-up table, a multiplier circuit and an output circuit consisting of a DAC (Digital-to-Analog Converter) and a low pass filter.

The data sent to the FMMS system hardware is controlled by an 8085 CPU. The CPU basically waits for interrupts from the keyboard port. When it senses that a key has been pressed or released, it sends out the appropriate data at the proper times to each of five generator registers.

1.2 Brief Historical Background

FM synthesis of music is a technique which is only about ten years old Synthesizing music by electronic means dates back to the early 1900's with the development of Thadeous Cahill's Telharmonium. This was a huge machine which was driven by alternators. In 1920, the Theremin was introduced by USSR's Leon Theremin. It was an antennae synthesizer which used the capacitance from one's hand to alter the beat frequency from dual oscillators. The first electronic music studios were introduced in the early 1950's.

With the development of the computer, new ways of synthesizing music seemed limitless. Max Mathews of Bell Labs was the first to use digital computers to produce music back in 1960. By the early 1970's, John Chowning demonstrated the effectiveness of frequency modulation techniques in simulating "natural instrument tones." The late 1970's marked the advent of fabricating music synthesizers on a single integrated circuit chip [1].

Presently, three means of synthesizing music are in vogue. Additive synthesis, which generates a sine wave for each harmonic and adds them all together. Non-linear synthesis, which "processes" an input sine wave into various harmonic frequencies. The processor essentially transforms the input into a power series expansion whose coefficients depend on the input amplitude [1]. Finally, FM, which controls the

evolution of the spectrum rather than controlling each individual frequency component.

CHAPTER 2

THEORETICAL CONCEPTS

2.1 Static Spectra

Many attempts have been made at producing tones with interesting timbres. Musical Engineers have tried using combinations of triangle, square, sine and sawtooth waveforms in an effort to simulate a timbre with more life and zest than the usual electronic sounding synthesizers which use square or sine waves exclusively. The problem with these attempts is that regardless of how one combines these various waveforms, the spectrum it produces will be static. For example, let us say one wants to produce an interesting tone with a fundamental frequency of 100Hz. By using a sine wave, the spectrum simply includes the fundamental at 100Hz as shown in Figure 2.1. Although simple, this tone lacks character. Next, one could use a square wave, producing the spectrum diagrammed in Figure 2.2. Square waves sound nasally and

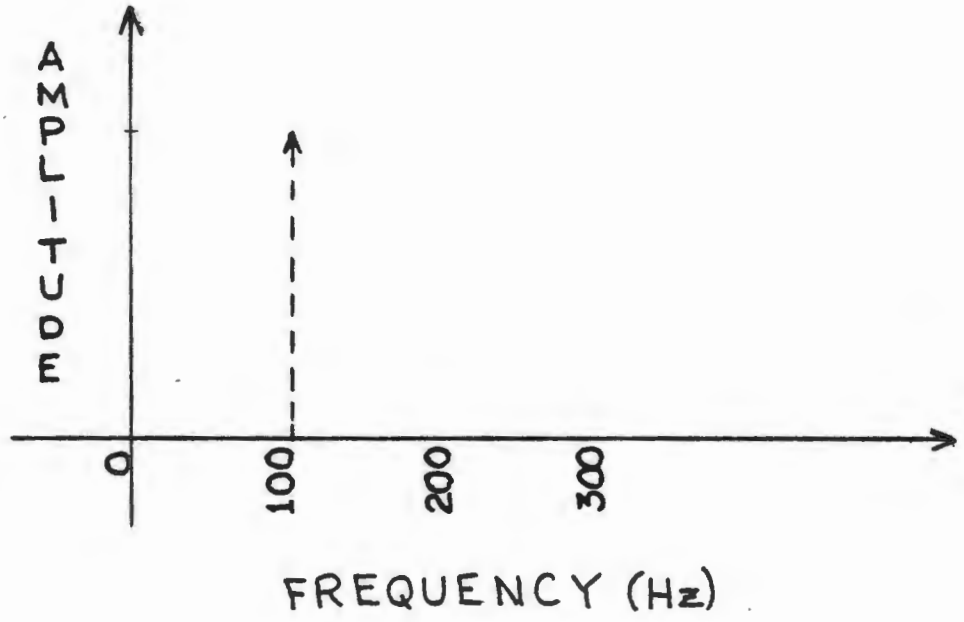


Figure 2.1 SINE WAVE SPECTRUM

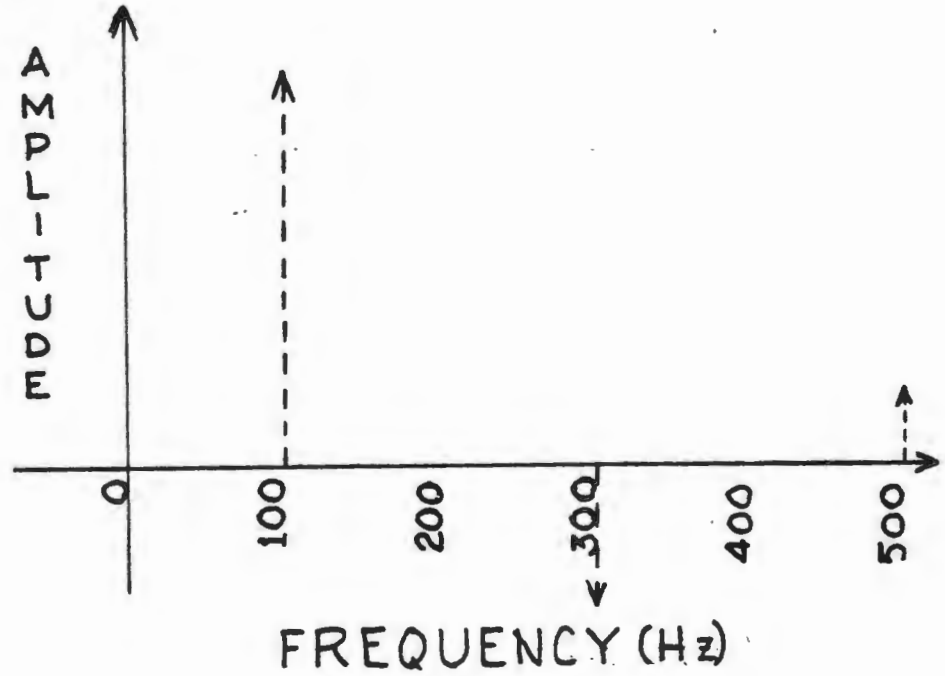


Figure 2.2 SQUARE WAVE SPECTRUM

somewhat piercing. In Figure 2.3, the previous spectrum is added to that of a sawtooth wave (with fundamental frequency of 100Hz).

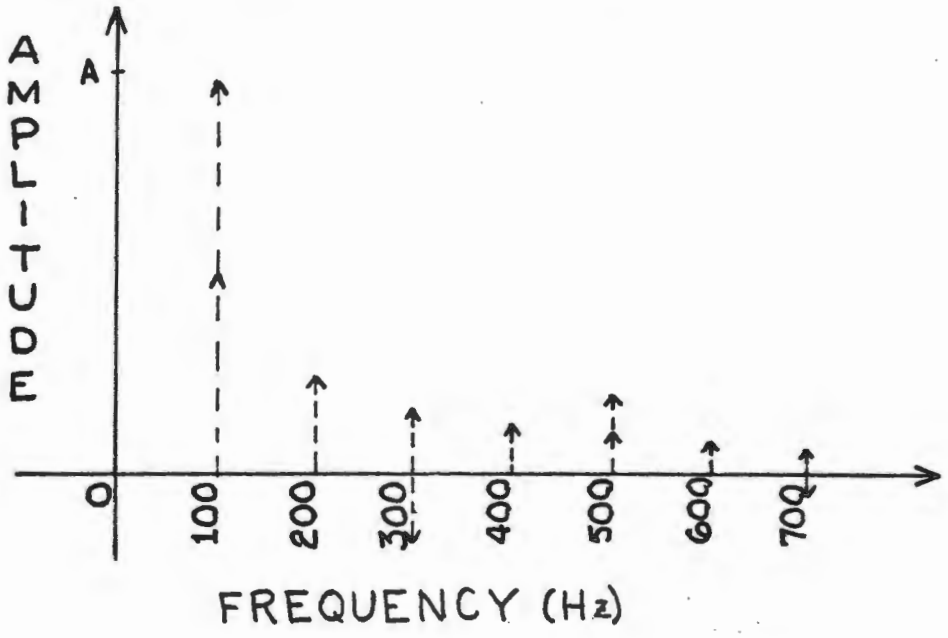


Figure 2.3 SPECTRUM OF SAWTOOTH AND SQUARE WAVE COMBINATION

By adding these components vectorially one notices that every other odd harmonic has been eliminated. This spectrum is becoming broader (amplitudes of harmonics are increasing). This tends to add more brightness or color to the tone.

The purpose of the previous illustrations is to give the reader some idea of what is meant by a broad spectrum and how a note's timbre can be enriched by broadening the spectrum.

Notice that in the last two illustrations, although the spectrum is complex, the amplitudes of all the frequency components remain constant throughout the duration of the note. It is for this reason that these techniques have failed in any attempts to simulate common musical instruments (e.g. brass, woodwinds, percussion etc.), the ultimate test of a music system's capabilities [2].

2.2 Dynamic Spectra

The FM equation allows dynamic control of the spectrum by varying a parameter called the modulation index. If the timbre of common instrument tones depended upon the relative magnitudes of each frequency component, the FM approach would probably be ineffective. However, the results achieved by John Chowning and others have shown that the timbre is more dependent on the evolution of the spectrum as a whole rather than the evolution of individual components [3].

2.2.1 Mathematical description

To see how a dynamic spectrum is realized using linear frequency modulation, the general FM equation will first be derived. To produce an FM signal, one superimposes an AC waveform on a DC waveform and uses this signal to control a variable oscillator whose instantaneous output frequency is proportional to the input signal. Thus:

$$f_{inst} = KS_{d.c} + KS_{a.c.}$$
$$= f_c + f_d \cos (2\pi f_m t) \quad (1)$$

where the modulating frequency, f_m, is the frequency of the AC signal. The carrier frequency, f_c, is proportional to the DC signal, and f_d is the maximum amount by which the output frequency deviates from the carrier frequency. The output signal could be thought of as the vertical projection of a vector rotating on the unit circle as illustrated in Figure 2.4. The function A(t) is the instantaneous phase of the vector S(t). Since the time derivative of the phase is the instantaneous frequency, an expression for the instantaneous phase can be obtained by integrating equation (1):

$$A(t) = 2\pi \int_0^t [f_c + f_d \cos (2\pi f_m t')]\, dt'$$
$$= 2\pi f_c t + (f_d/f_m) \sin (2\pi f_m t) \quad (2)$$

Therefore the output signal S(t) is:

$$S(t) = \sin [2\pi f_c t + (f_d/f_m) \sin (2\pi f_m t)] \quad (3)$$

The coefficient f_d/f_m is known as the modulation index (M). Equation (1) can now be rewritten in the following form:

$$f(n) = f_c + (Mf_d) \cos (2\pi f_m nT) \quad (4)$$

Where equation (1) has been changed from a continuous to a discrete frequency whose sampling period is denoted by T. The output signal, therefore, can be completely specified by the three parameters: f_c, f_m and M.

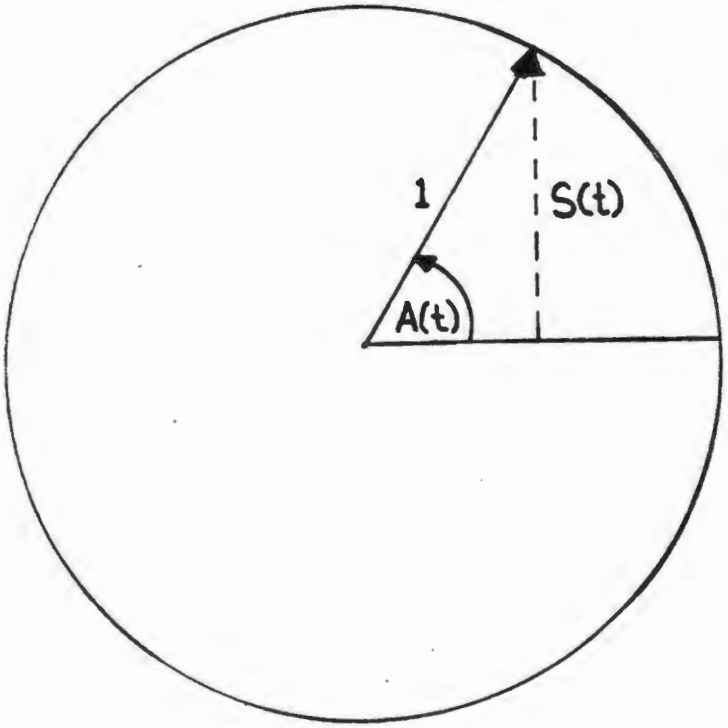


Figure 2.4 ROTATING VECTOR REPRESENTATION OF FM SIGNAL

Now by referring back to equation (3) we can begin our realization of the dynamic spectral behavior that is characteristic of frequency modulation.

Using a trigonometric identity:

$$S(t) = \sin(2\pi f_c t)\cos[M\sin(2\pi f_m t)] + \cos(2\pi f_c t)\sin[M\sin(2\pi f_m t)] \quad (5)$$

In order to simplify this expression still further one needs to utilize the "Bessel Functions" which define the sine of a sine and cosine of a sine in terms of Bessel polynomials. The identities are:

$$\sin[M\sin(x)] = 2[J_1(M)\sin(x) + J_3(M)\sin(3x) + \ldots] \quad (6)$$

$$\cos[M\sin(x)] = J_o(M) + 2[J_2(M)\cos(2x) + J_4(M)\cos(4x) + \ldots] \quad (7)$$

Thus:

$$S(t) = \sin(2\pi f_c t)\, J_o(M) + 2[J_2(M)\cos(4\pi f_m t) + \ldots] + \cos(2\pi f_c t)\, 2[J_1(M)\sin(2\pi f_m t) + J_3(M)\sin(6\pi f_m t) + \ldots] \quad (8)$$

Again using trigonometeric identities:

$$\begin{aligned} S(t) = {} & J_o(M)\sin(2\pi f_c t) \\ & + J_1(M)\sin[2\pi(f_c + f_m)]t - \sin[2\pi(f_c - f_m)]t \\ & + J_2(M)\sin[2\pi(f_c + 2f_m)]t + \sin[2\pi(f_c - 2f_m)]t \\ & + J_3(M)\sin[2\pi(f_c + 3f_m)]t - \sin[2\pi(f_c - 3f_m)]t \end{aligned} \quad (9)$$

2.2.2 Example

Now to illustrate spectral evolution let us take an example. Suppose f_c = 200Hz, f_m = 100Hz and f_d = 100Hz (M=1). The Bessel Function values for M=1 are: J_o = .77; J_1 = .44; J_2 = .11; J_3 = .02. This will produce the spectrum illustrated in Figure 2.5. If the amount of frequency deviation is increased from 100Hz to 300Hz (M=3), the result shown in Figure 2.6 is obtained.

Since sin(-x) = -sin(x), the negative frequency components get reflected and add vectorially to the positive frequency components. One will notice, however, that the spectrum has "spread out" yielding higher frequency components of significant weight. Essentially, some of the energy from the carrier signal has been distributed to the other higher and lower frequency components. This, finally, is what is meant by spectrum evolution. As the spectrum continues to broaden, the timbre of the note gets richer and more interesting [4].

One can control this dynamic action of the spectrum by simply applying a time envelope on the modulation index parameter. This technique is used in the FMMS. In addition, the FMMS has another time envelope which is applied to the amplitude of the output wave yielding this final FM equation:

$$S(nt) = A(nt) \sin [2\pi f_c nt + M(nt) \sin (2\pi f_m nt)] \qquad (10)$$

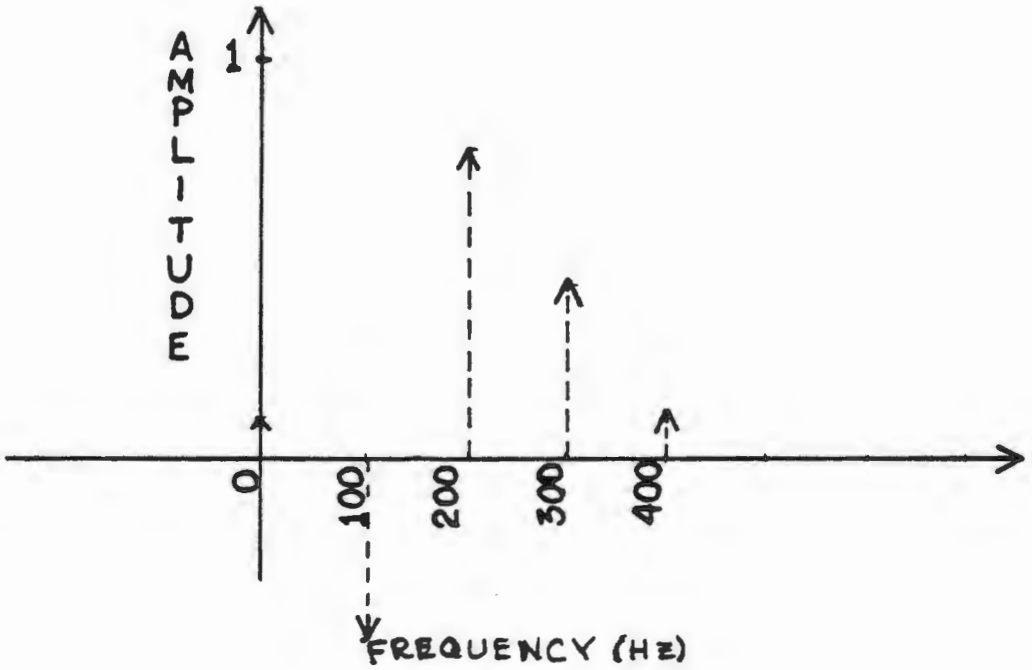


Figure 2.5 FM SPECTRUM (M=1)

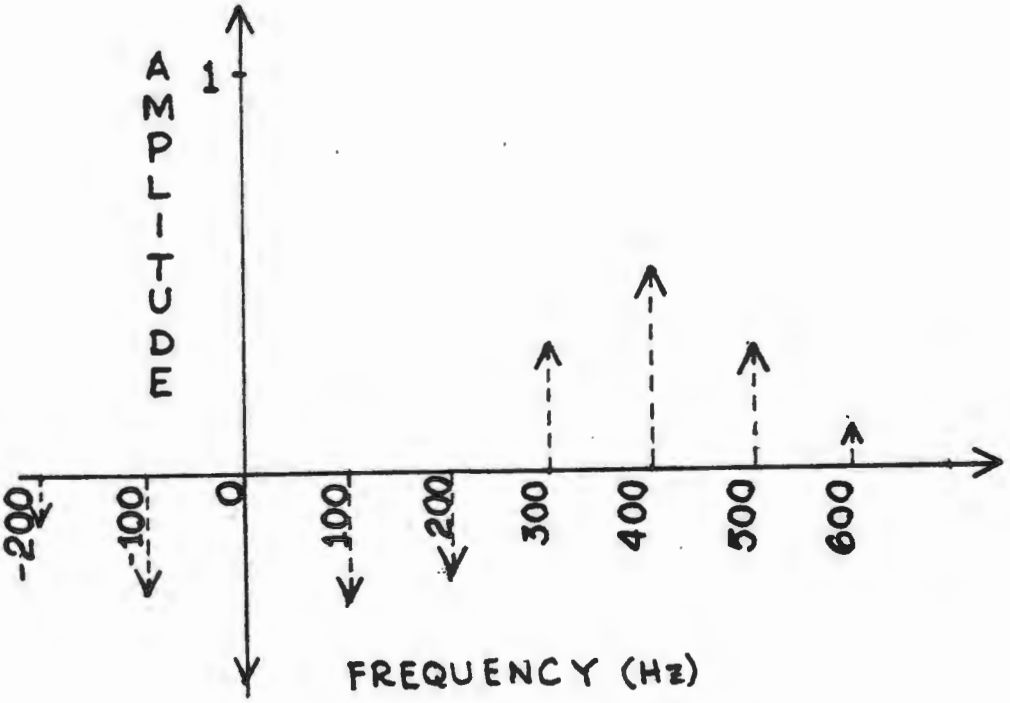


Figure 2.6 FM SPECTRUM (M=3)

Each instrument, therefore, will have three controlling parameters: A(nT), the output amplitude envelope function; M(nT), the modulation index envelope function and (f_c/f_m), the ratio of the carrier frequency to the modulating frequency. If one represents this ratio in the form of N_1/N_2, where N_1 and N_2 have all common factors divided out, then the fundamental frequency of the modulated carrier will be:

$$f_o = f_c /N_1 = f_m /N_2 \qquad (11)$$

By making the ratio of f_c to f_m a small integer, a harmonic spectrum results (as can be seen in the example). However, if this ratio is non-integral, an inharmonic spectrum results, which should be useful in simulating percussive sounds.

CHAPTER 3

SYSTEM DESIGN

This and the next chapter deal with the hardware design of the FMMS. This chapter describes each subsystem in block diagram form and then puts the units together to give the reader a view of the overall system. Most of the specifics in the design are presented in chapter four.

In order to realize equation (10) as a digital hardware system, one needs to have a modulating frequency generator, a carrier frequency generator, a multiplier, an amplitude and modulation index envelope generators and a DAC.

3.1 Frequency Generators

The frequency generators are essentially phase angle generators. The idea is to load a register with a value (phase angle increment) which is proportional to the frequency one wants generated. This value gets periodically added into an accumulator register whose output is used to address a sine wave table. When the accumulator overflows, a new cycle of the sine wave begins. Thus, the number of times the accumulator overflows per second, will be the frequency of the output sine wave. Loading the increment register with a larger value will cause the accumulator to overflow more often and thus increase the frequency of the output wave. This scheme is illustrated in Figure 3.1.

The output frequency is dependent on the phase angle increment (I), the sampling rate (S), and the sine table length (L), by the following equation:

$$f = I \times (S/L) \qquad (12)$$

These three values (I,S and L) depend on how elaborate a system one wants. The considerations made in determining them can be found in the next chapter. In order to increase the utilization of the sine wave table, the ROM (Read Only Memory) address was multiplexed between the modulating phase angle and the modulated carrier phase angle.

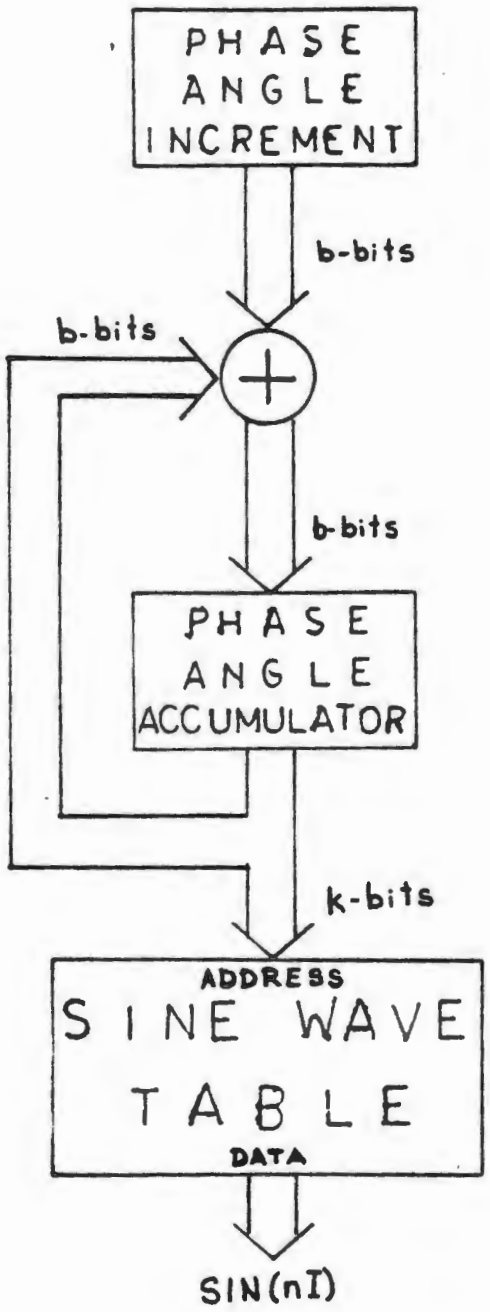


Figure 3.1 FREQUENCY GENERATOR BLOCK DIAGRAM

3.2 Envelope Generators

The envelopes generated by the FMMS are concatenated linear segments whose slopes, which are preprogrammed, are loaded into a slope increment register. Like the frequency generators, this increment gets added periodically into an accumulator register. Unlike the frequency generators, however, where overflow starts a new cycle of the waveform, the envelope accumulator overflow simply increments or decrements a ramp register. Figure 3.2 illustrates this process. This design technique was used to shorten the register lengths which would need to be twice as large if the technique illustrated in Figure 3.1 were used.

Once again, the value loaded into the slope increment register depends on the sampling frequency and its value should be adjusted so that it never causes overflow in the ramp register. For the duration of the note, the CPU is responsible for updating the slope increment register to alter the slope of the ramp. M(nT) and A(nT) are generated in this part of the system.

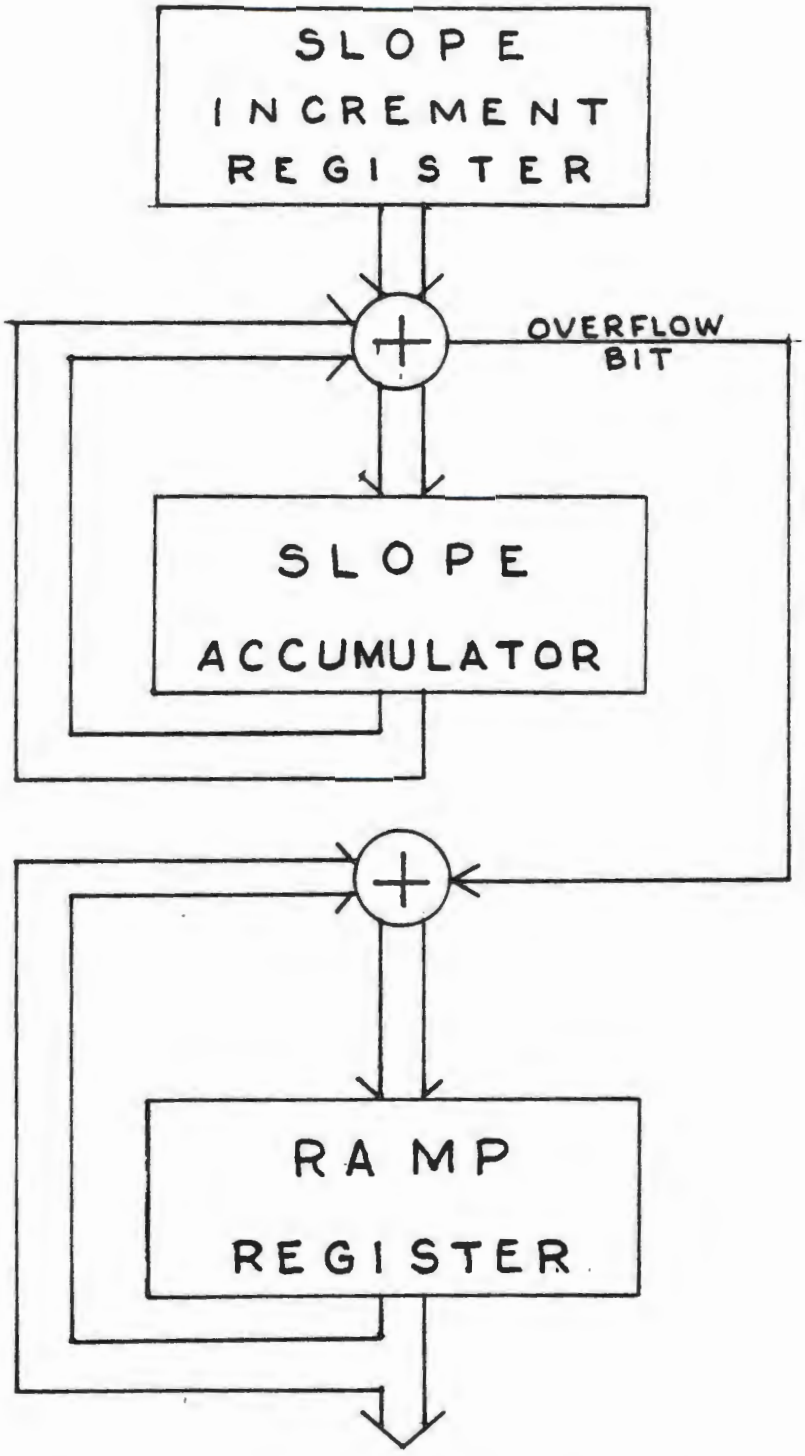


Figure 3.2 ENVELOPE GENERATOR BLOCK DIAGRAM

3.3 Multiplier Circuit

In order to provide for the dynamic spectrum, the modulating frequency needs to be multiplied by a varying modulation index M(nT) (provided by the envelope generator subsystem). In addition, the modulated carrier wave is multiplied by A(nT) to provide for a dynamic output amplitude. These two functions were multiplexed into the multiplier input of a multiplier, and the output of the sine wave generator was used as the multiplicand. As one can begin to discern, the timing is very critical. Refer to the timing diagram in APPENDIX B for the complete details on the timing of the pipeline.

3.4 Digital-to Analog Conversion

Once each sample has been through the pipeline twice (once in forming the instantaneous modulation frequency and the second time forming the instantaneous modulated carrier frequency), it is sent to the output register which converts this digital data into an analog voltage. Then, after going through a low pass filter, it is sent out to the Music System amplifier and speaker. A 16-bit DAC would produce very low distortion, however it would cost hundreds of dollars. Therefore a 12-bit DAC was used, which for the FMMS is quite adequate.

Figure 3.3 shows a block diagram of the entire FMMS. In the next chapter, specifics concerning register length, sampling frequency, type of multiplier and sine wave table length are discussed.

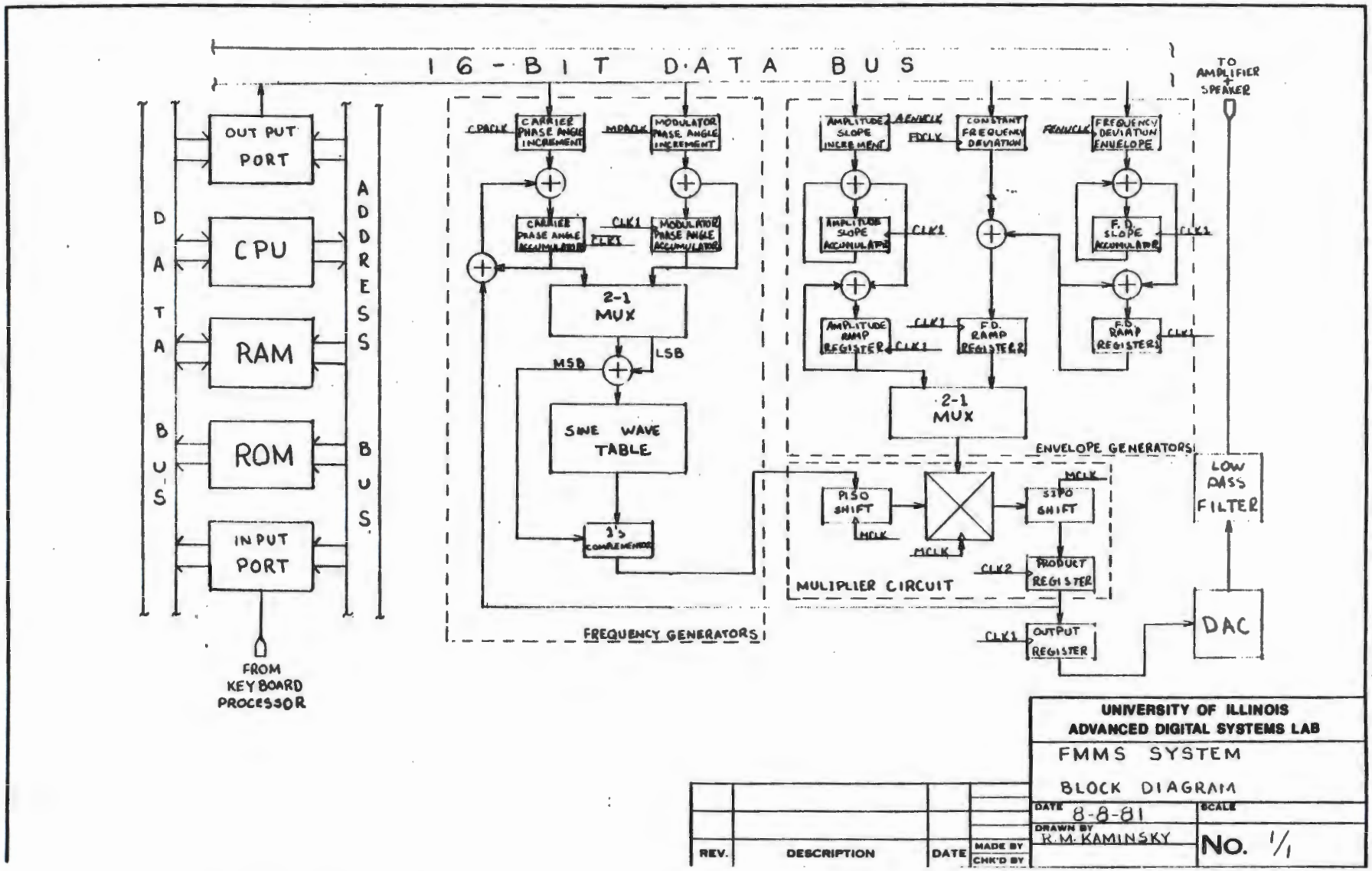


Figure 3.3 FMMS BLOCK DIAGRAM

CHAPTER 4

HARDWARE DESIGN

The previous chapter illustrated the general hardware structure of the FMMS. This chapter describes the various considerations which were made in completing the design of the hardware.

4.1 Register Length

The first concern was how large to make the pipeline registers in the frequency generator and envelope generator circuits.

4.1.1 Frequency generator circuit

For the frequency generators, the register length depends on the maximum and minimum frequencies one would like to generate. Since the FMMS is to interface with the organ keyboard, the maximum allowable frequency should be 2^{13}-1Hz. An additional sign bit is needed since frequency modulation requires negative as well as positive frequencies.

To determine the minimum frequency, one must weigh cost as well as resolution. The most noticeable "frequency jitter" will be heard in the lower frequencies. Generally, comparisons are done around note C_2 = 65.4Hz. The unit of measure is the cent (one cent = 1/1200 of an octave). Therefore:

$$1 \text{ cent} = 2^{1/1200} \qquad (13)$$

To form a basis for comparison, some of the most elaborate systems have a frequency resolution of one cent at 65.4Hz. Expensive tape recorders have wow and flutter figures around 0.1% or lower. At 65.4Hz, this would be equivalent to 1.73 cents. In Table 4.1, cent deviation comparisons are made for registers having 14, 16 and 18 bits.

Table 4.1 ACCURACIES OF VARIOUS REGISTER LENGTHS

# of bits	f(min)	f(max)	cents dev. around 65.4Hz
14	1.0000	8191	26.30
16	0.2500	8191	6.60
18	0.0625	8191	1.65

Obviously, with 18-bit registers, one would be able to produce the kind of accuracy available on some of the most expensive systems. However to do this would require an 8-bit microprocessor to do triple precision arithmetic. Microprocessors are relatively slow at performing data manipulation in music synthesis applications, so this slowdown would be more detrimental than would the loss of accuracy in using a smaller register length. For this reason the FMMS uses 16-bit pipeline registers in the frequency generator circuit.

4.1.2 Envelope generator circuit

The decision on how large to make the envelope generator registers depends on how much noise can be tolerated after the multiplier circuit. Figure 4.1 illustrates the model used in evaluating the SNR (Signal-to-Noise Ratio) for the signal coming out of the multiplier.

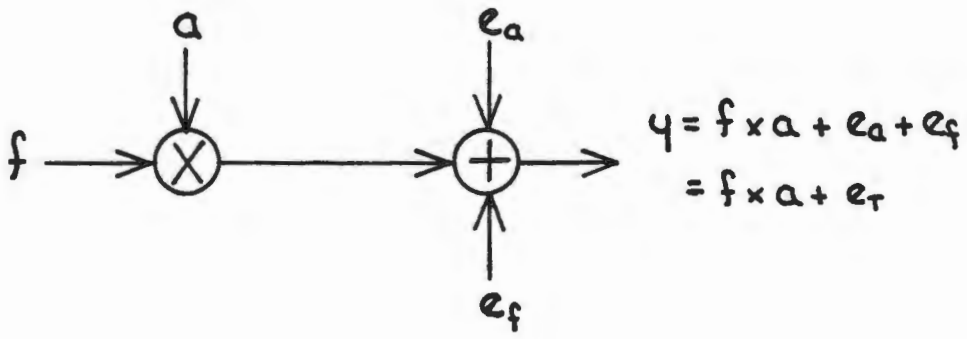


Figure 4.1 MODEL OF MULTIPLIER QUANTIZATION NOISE

The errors are due to truncation of the frequency and amplitude signals. For this calculation, the two input signals are considered to be fractions, therefore the error will be equivalent to the weight of the least significant bit position. As an example, consider f & a to be 8 bits each (including sign), then:

$$e_a + e_f = 2^{-7} + 2^{-7} = 2^{-6} = e_T$$
$$\text{SNR} = 10 \log (1/e_T^2) = 36.12\text{dB} \qquad (14)$$

where the variance of the signal is normalized to 1. Table 4.2 lists the SNR's for several amplitude and frequency register lengths.

Table 4.2 MULTIPLIER OUTPUT SNR FOR VARIOUS INPUT REGISTER LENGTHS

f (bits)	a (bits)	SNR (dB)
12	8	41.62
12	12	60.21
12	16	65.70
16	8	42.11
16	12	65.70
16	12	65.70
12	INFINITE	66.23
16	INFINITE	90.31

The 8-bit register values appear rather poor. A SNR around 60dB would be much better. For this reason, 12-bit registers are used in the envelope generator circuit.

4.2 Pipeline Clock

The next concern was the frequency at which to sample the data. By having the pipeline clock be a power of 2, the frequency increment values would be equal to the actual frequency desired, but shifted right or left. Clock frequency values of 2^{14} or 2^{15} were possibilities. By making it much higher, one decreases the throughput of the pipeline. To be able to completely recreate the original waveform, the sampling frequency should be at least 2 1/2 times the highest frequency component. This requirement is not met using a clock frequency of 2^{14}Hz since the highest fundamental frequency is 2^{13}Hz. Therefore, $f_{clk} = 2^{15}$Hz in the FMMS.

4.3 Sine Wave Table

A sine wave table stored in ROM is probably the easiest way to convert an instantaneous phase into the sine function. Again, a decision must be made as to how long the table should be and how many bits each word in the table should have.

In order to properly evaluate what size table should be used, reference was made to an article written by Richard Moore entitled "Table Lookup Noise for Sinusoidal Digital Oscillators" [5]. In the article, Moore compares the worst-case SN_eR (signal-to-error noise ratio) of various size tables using three methods of addressing the table: truncation, rounding and interpolation.

A study conducted at the Center for Computer Research in Music and Acoustics at Stanford University concluded that a 4096-word x 12-bit/word table would produce very negligible distortion [6]. This table, however, contained 360 degrees of the sine wave and truncated the lower address bits. Moore's tables contain only 180 degrees of the sine wave and therefore use the most significant address bit to determine the polarity of the sine wave. Thus, the table size mentioned above would be equivalent to a 2048-word x 11-bit/word table which Moore lists as having a SN_eR of 54.4dB. The ADSL has an abundance of 2K x 8 EPROM's (Eraseable-Programmable Read Only Memory) (2716) so two of these chips were used to build a 2K x 15-bit sine wave table. If truncation were used, this would yield a SN_eR of 54.4dB (according to the article, adding more than b bits, where 2^b = memory length, does not improve the SN_eR since then practically all the noise is due to the finite table length). By adding just three more adder chips a round-off adder was inserted, thus improving the SN_eR to 60.4dB.

4.4 Multiplier Circuit

The multiplier circuit, which is the bottleneck of the pipeline, has to be fast enough to perform six multiplications in one-quarter of a pipeline cycle if the FMMS is to be expanded to six channels. This requires each multiplication to be completed in 1.27 microseconds. A cheap, but slow serial multiplier that can perform 2's complement, 8x8 multiplication is the AMD25LS14. It is expandable and can perform a 16- x 12-bit multiply in less than 1.2 microseconds, and costs less than $10 per chip. This chip works very well in the FMMS, although it does require some additional hardware to perform synchronization and clocking. (Refer to Figure A10 for the multiplier synchronization schematic.)

CHAPTER 5

SYSTEM SOFTWARE

This chapter deals with the software in the FMMS. The major task of the software is to provide dynamic control of the envelope generators, and supervise the inputs from the ADSL Music System's keyboard. Two programs will be described. The first is a test program which allows the user to experiment with different sounds by allowing him to write the various FM parameters directly into RAM (Random Access Memory) through the use of a monitor program (refer to MUSMON monitor listing in APPENDIX D). The second description is for a program which interfaces the FMMS directly to the keyboard port. This program allows the user to play the FMMS in real time. The parameters for the particular instrument being played will be stored in ROM.

5.1 Test Program

5.1.1 Parameters

For each note that is played, five registers have to be loaded with values related to the type of sound being synthesized. These registers consist of the modulating phase angle increment (MPA), the carrier phase angle increment (CPA), the constant frequency deviation (FD), the frequency deviation envelope (FENV) and the amplitude envelope (AENV).

5.1.1.1 Scaling. The registers in the frequency generator circuit contain 16 bits (2 fraction bits, 13 integer bits and 1 sign bit). Therefore the value which is loaded into MPA and CPA will be the desired frequency scaled up by four. For instance, a frequency of 1024Hz will have a phase angle increment value of 4096. Keep in mind that the fundamental frequency of the modulated carrier will depend on the ratio of f_c to f_m (refer back to chapter two).

The registers in the envelope generating circuit contain only 12 bits (11 magnitude bits and 1 sign bit). Since the value loaded into the frequency deviation registers is the product of the modulating frequency and the modulation index, this value will need to be scaled down. The maximum positive slope increment is $2^{11}-1$. This value will therefore represent the maximum modulation frequency ($2^{13}-1$) times the

maximum modulation index (8), for a segment with maximum slope [(max amp change)/(min seg duration) = 1/(1/8) = 8]. This product yields $2^{19}-64$, therefore each value must be right-shifted by 8 places. The maximum possible value for the frequency deviation is $8x(2^{13}-1)$ or $2^{16}-8$. Therefore this value must be right-shifted by 5 places.

Since each envelope consists of four segments (representing the attack, dacay, sustain and release of the note), eight parameter values will be needed for each envelope. Each segment has an increment value (which may be negative) along with a duration value. The CPU uses the duration value as a counter to signal when a new segment increment value should be loaded.

5.1.1.2 Timing. In order to properly scale each instantaneous frequency, the parameters must be loaded into their respective pipeline register during the correct clock cycle. To do this, the CPU samples the state of the pipeline clock and after the appropriate number of cycles have elapsed, loads the next parameter into its corresponding pipeline register.

5.1.2 Program description

The test program allows the user to experiment with different sounds by programming his own values for the parameters listed in Table 5.1, into the FMMS. (Refer to APENDIX C for program listing and sample test data.)

5.2 Keyboard Interface Program

In order to play music in real time, an input port on the FMMS is connected to an output port from the ADSL Music System keyboard processor board. When the keyboard processor detects that a note has been pressed or released, it sends an interrupt to the FMMS processor. The FMMS processor then checks the code in the input port. If it is all zeros it jumps to the released key routine. If not, it splits the code into a lower and upper nibble. It then takes the lower nibble (which is the code for the pitch of the note) and jumps to a table in ROM containing all the parameters for that particular pitch in its highest octave. Then it takes the upper nibble (containing a code for the octave of the note) and divides the frequency dependent parameters in the table by a number which will bring the note into the proper octave. These modified parameter values are then stored in a buffer.

Table 5.1 TEST PROGRAM PARAMETERS

PARAMETER	RANGE	DESCRIPTION
FC	0-7FFFH	CARRIER PHASE ANGLE INCREMENT
FM	0-7FFFH	MODULATION PHASE ANGLE INCREMENT
FD	0-07FFH	FREQUENCY DEVIATION
FENV1-4	0-07FFH (POS) 0FFF-0801H (NEG)	FREQUENCY DEVIATION ENVELOPE INCREMENT SEGMENTS 1-4
AENV1-4	0-07FFH (POS) 0FFF-0801H (NEG)	AMPLITUDE ENVELOPE INCREMENT SEGMENTS 1-4
SD1-4	01-FF	ENVELOPE SEGMENT DURATION

After the decoding, scaling and storing have been completed, the processor sends the parameters to a routine similar to the Test Program discussed in the previous section. For the third envelope segment (sustain), the processor loads a zero increment into the amplitude and frequency deviation envelope registers. This will cause them to hold their ramp registers constant. The processor then waits for a released key interrupt. Upon receipt of the interrupt, it sends out the last segment increment value (release), to the envelope registers. After the duration of the release segment, the processor clears all pipeline registers and waits for another keypress.

CHAPTER 6

CONCLUSIONS

6.1 Future Expansion

The present version of the FMMS is single-channel. However, it represents the basic building block upon which larger, more sophisticated systems can be constructed. In the following paragraphs, a few suggestions are given on how the FMMS may be expanded into a multivoice system.

In order to generate multiple, independent, output waveforms, the existing hardware will need to be time-multiplexed. The limiting factor as to how many additional channels may be added will be the speed of the multiplying pipeline segment. In order to perform the multiplexing, each of the increment registers will write their values into a RAM. Once the RAM's have been loaded, a special counter circuit steps through

the memory, reading both the increment and the accumulated values. These values are then sent to an adder and the result is written back into the RAM accumulator location. This concept is illustrated in Figure 6.1. The special counting circuit has logic which executes the reads and writes at the proper times as well as clocking the holding registers.

In addition, the output circuit would require a separate output section for each channel. This is shown in Figure 6.2 for a system having six channels. A six-bit shift register, clocked at six times the pipeline frequency could be used to trigger the sample and hold (S/H) circuits.

6.2 Closing Remarks

The frequency modulation technique used in the FMMS appears to have enough versatility to be able to simulate common musical instrument timbres. More research needs to be done, however, on how the spectrum of different instruments evolves before one can program the FMMS parameters to "accurately" simulate such tones. Strange, yet interesting sounds are easy to produce through experimentation with the Test Program. The FMMS gives an added dimension to music composition in general and particularly to the ADSL Music System.

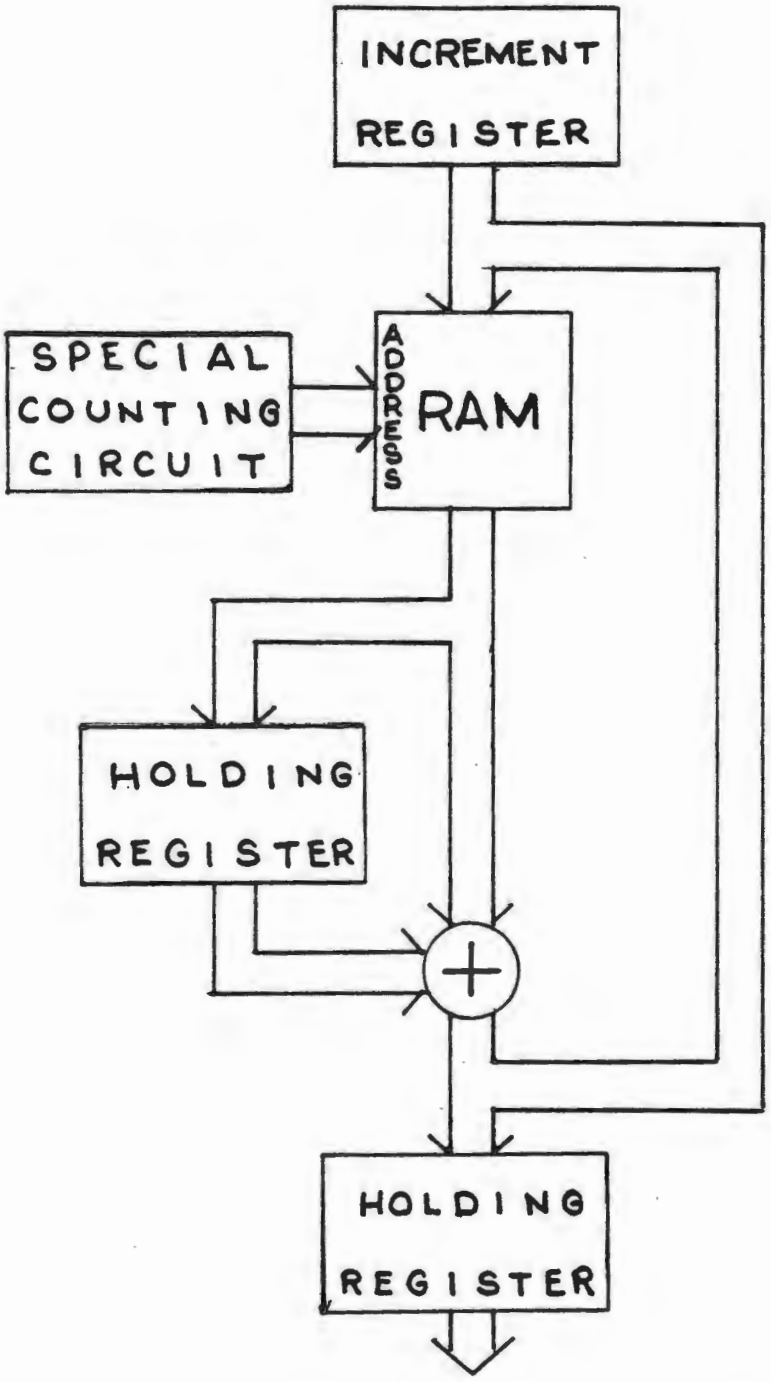


Figure 6.1 MODIFIED INCREMENT REGISTER FOR HANDLING MULTIVOICE

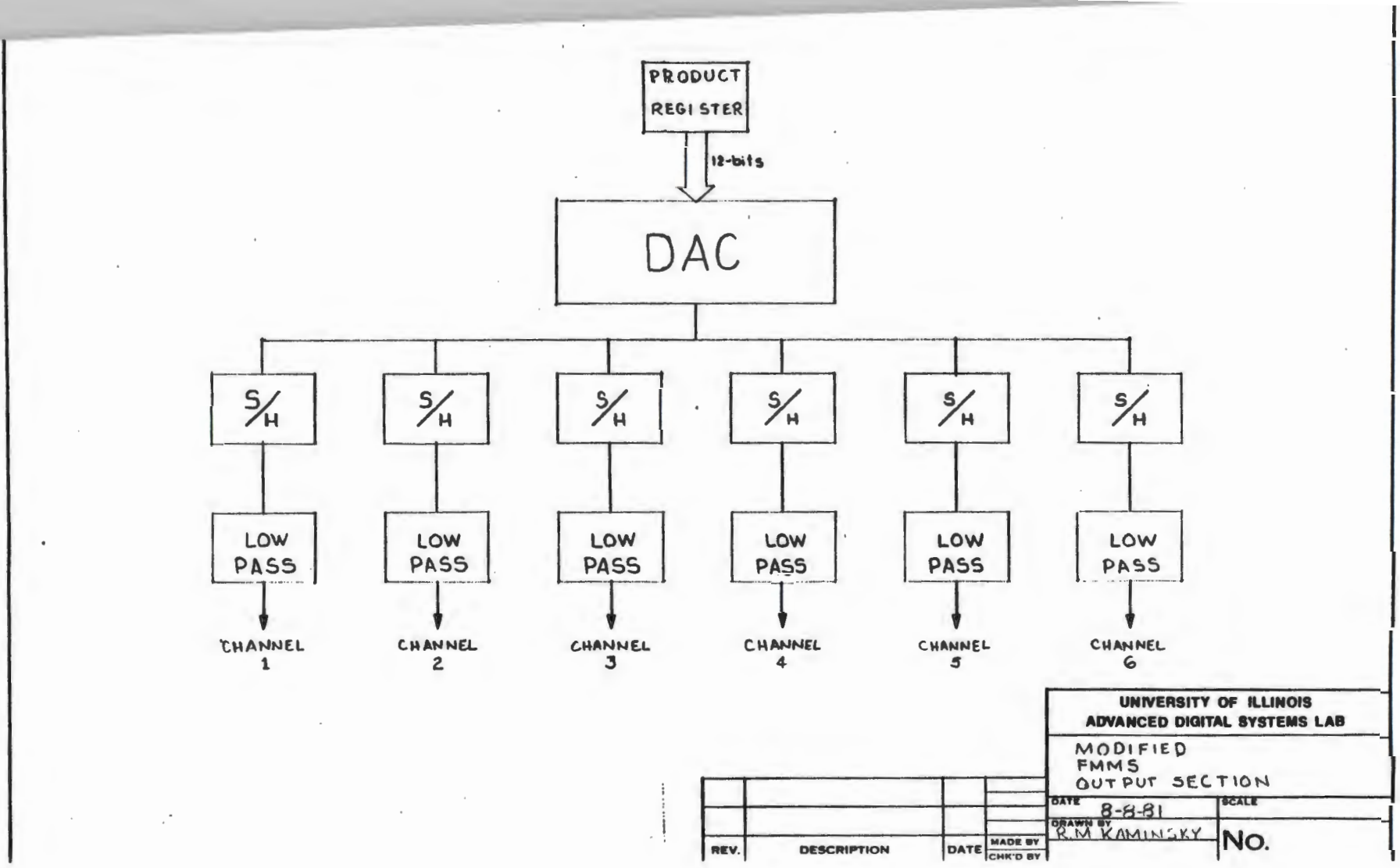
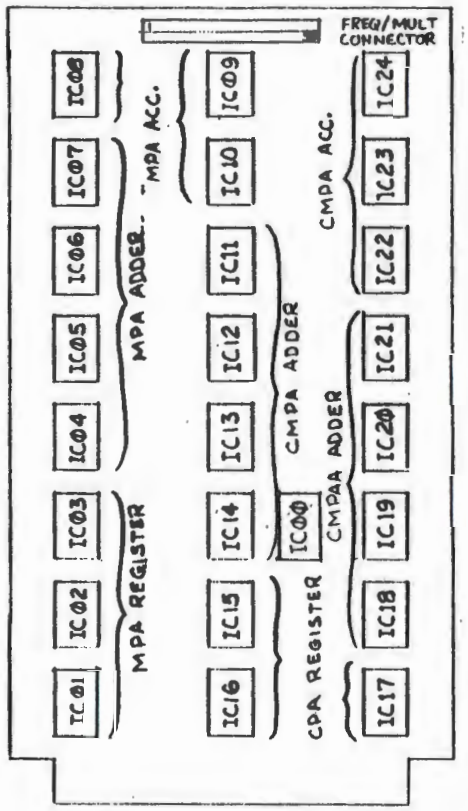


Figure 6.2 MODIFIED OUTPUT SECTION FOR HANDLING MULTIVOICE

APPENDIX A

FMMS HARDWARE SCHEMATICS



PIN	SIGNAL	PIN	SIGNAL
1	+5V	A	GND
2	NC	B	NC
3	NC	C	NC
4	NC	D	NC
5	NC	E	NC
6	NC	F	NC
7	NC	H	NC
8	NC	J	NC
9	NC	K	NC
10	NC	L	NC
11	NC	M	CPACLK
12	CLK1	N	MPACLK
13	$\overline{CLR}$	P	NC
14	DB14	R	DB15
15	DB12	S	DB13
16	DB10	T	DB11
17	DB08	U	DB09
18	DB06	V	DB07
19	DB04	W	DB05
20	DB02	X	DB03
21	DB00	Y	DB01
22	GND	Z	+5V

SYSTEM BUS CONNECTOR

PIN	SIGNAL	PIN	SIGNAL
1	MPA03	2	MPA04
3	MPA05	4	MPA06
5	MPA07	6	MPA08
7	MPA09	8	MPA10
9	MPA11	10	MPA12
11	MPA13	12	MPA14
13	MPA15	14	NC
15	CMPA03	16	CMPA04
17	CMPA05	18	CMPA06
19	CMPA07	20	CMPA08
21	CMPA09	22	CMPA10
23	CMPA11	24	CMPA12
25	CMPA13	26	CMPA14
27	CMPA15	28	NC
29	NC	30	NC
31	NC	32	NC
33	NC	34	NC
35	MSWA15	36	MSWA14
37	MSWA13	38	MSWA12
39	MSWA11	40	MSWA10
41	MSWA09	42	MSWA08
43	MSWA07	44	MSWA06
45	MSWA05	46	MSWA04
47	MSWA03	48	MSWA02
49	MSWA01	50	MSWA00

FREQ/MULT CONNECTOR

DESCRIPTION OF ACRONYMS

MPA— MODULATOR PHASE ANGLE
CMPA— CARRIER+MODULATOR PHASE ANGLE
CMPAA— CMPA ACCUMULATOR
CPA — CARRIER PHASE ANGLE
FREQ/MULT— FREQUENCY BOARD TO MULTIPLIER BOARD
GND — GROUND
CLK — CLOCK
CLR — CLEAR
DB — DATA BUS
MSWA— MODULATION INDEX TIMES SINE WAVE AMPLITUDE

REV.	DESCRIPTION	DATE	MADE BY / CHK'D BY

UNIVERSITY OF ILLINOIS
ADVANCED DIGITAL SYSTEMS LAB

FREQUENCY GENERATOR BOARD

COMPONENT LAYOUT & CONNECTOR LIST

DATE 8-8-81 | SCALE

DRAWN BY R.M. KAMINSKY | NO. 1 OF 3

Figure A1 COMPONENT LAYOUT FOR FREQUENCY GENERATOR BOARD

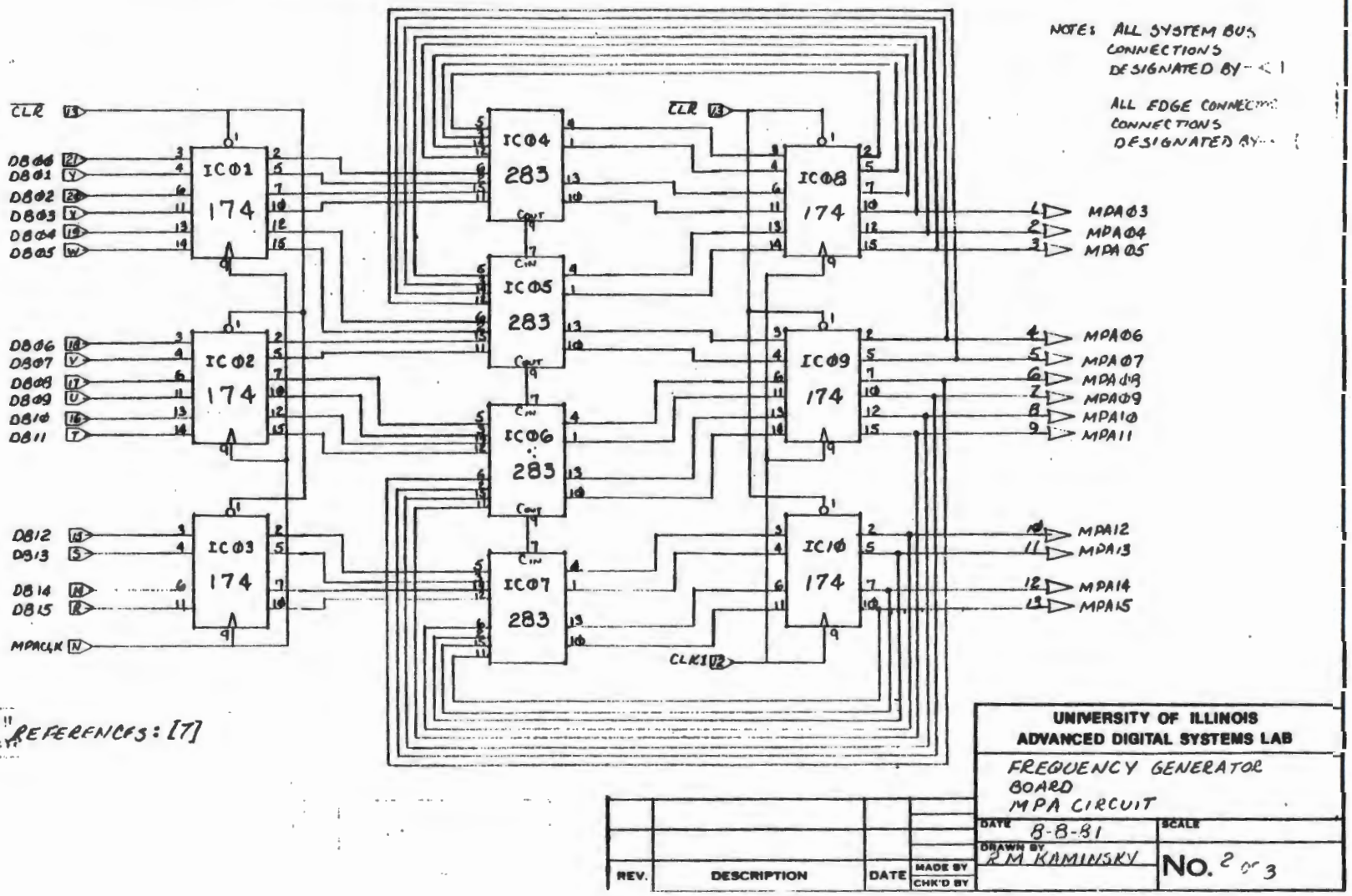


Figure A2 MPA CIRCUIT

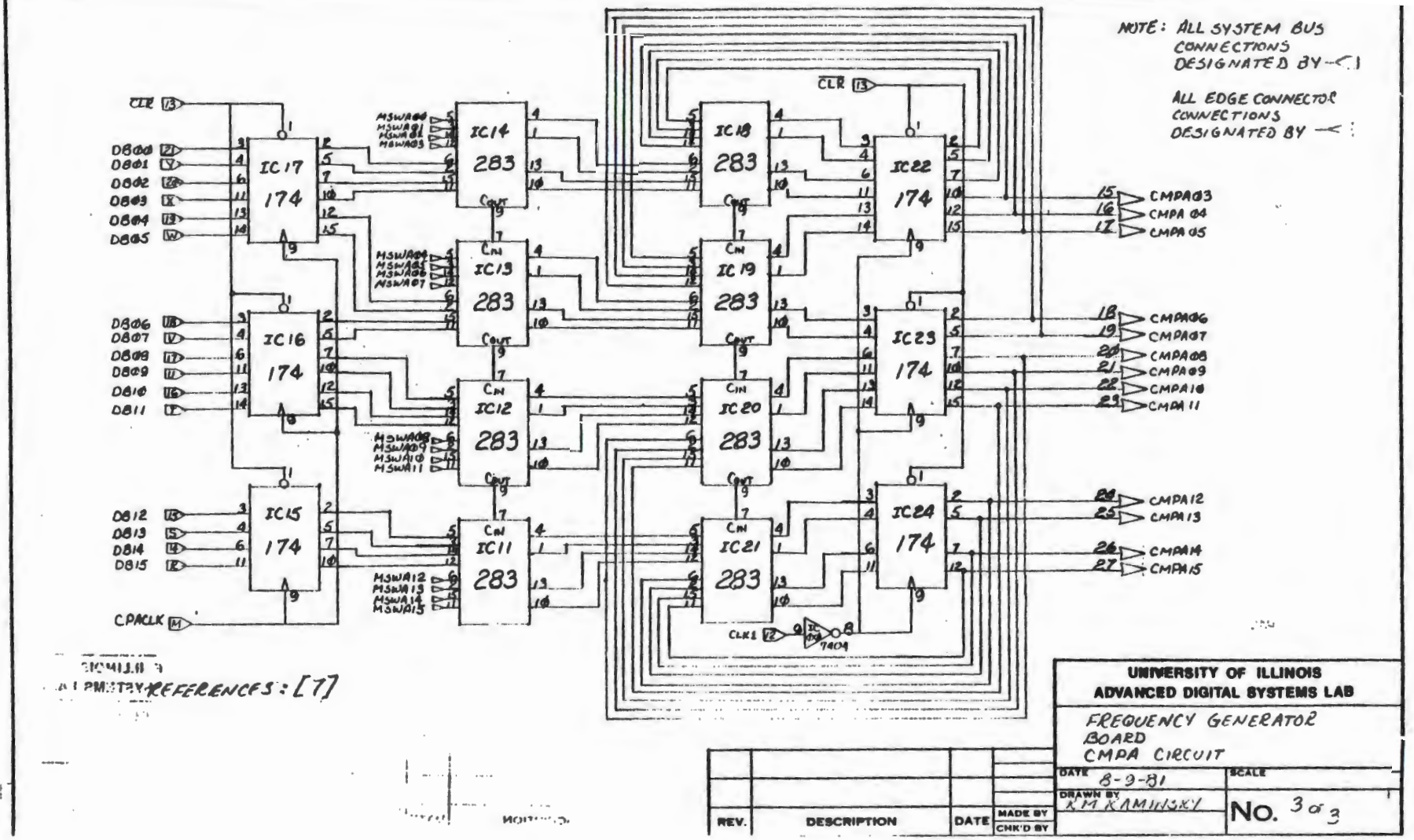
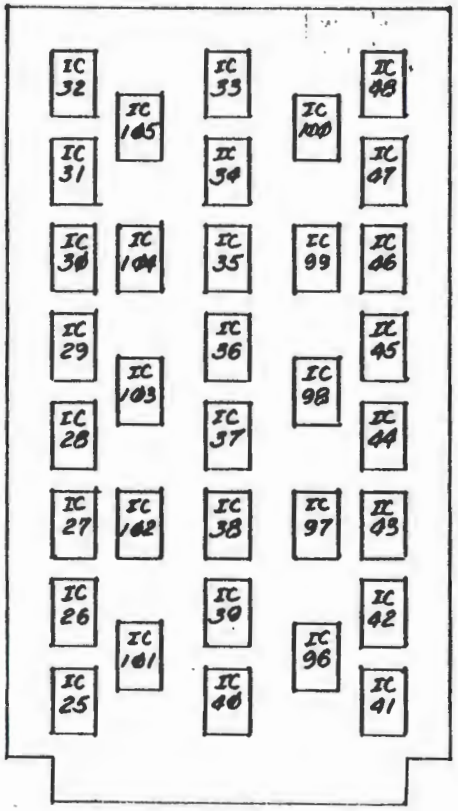


Figure A3 CMPA CIRCUIT



WIRING SIDE

PIN	SIGNAL	PIN	SIGNAL
1	+5V	A	GND
2	MCND01	B	MCND00
3	MCND03	C	MCND02
4	MCND05	D	MCND04
5	MCND07	E	MCND06
6	MCND09	F	MCND08
7	MCND11	H	MCND10
8	NC	J	NC
9	NC	K	NC
10	RENRLK	L	NC
11	FCCLK	M	NC
12	CLK1	N	NC
13	CLR	P	AENRLK
14	NC	R	NC
15	NC	S	NC
16	DB10	T	DB11
17	DB08	U	DB09
18	DB06	V	DB07
19	DB04	W	DB05
20	DB02	X	DB03
21	DB00	Y	DB01
22	GND	Z	+5V

SYSTEM BUS CONNECTOR

DESCRIPTION OF ACRONYMS

MCND- MULTIPLICAND

FENV - FREQUENCY DEVIATION ENVELOPE

AENV- AMPLITUDE ENVELOPE

FD - FREQUENCY DEVIATION

REV.	DESCRIPTION	DATE	MADE BY / CHK'D BY

UNIVERSITY OF ILLINOIS
ADVANCED DIGITAL SYSTEMS LAB

ENVELOPE GENERATOR BOARD
COMPONENT LAYOUT AND CONN. LIST

DATE 8-9-81 | SCALE

DRAWN BY R M KAMINSKY | No. 1 OF 4

Figure A4 COMPONENT LAYOUT FOR ENVELOPE GENERATOR BOARD

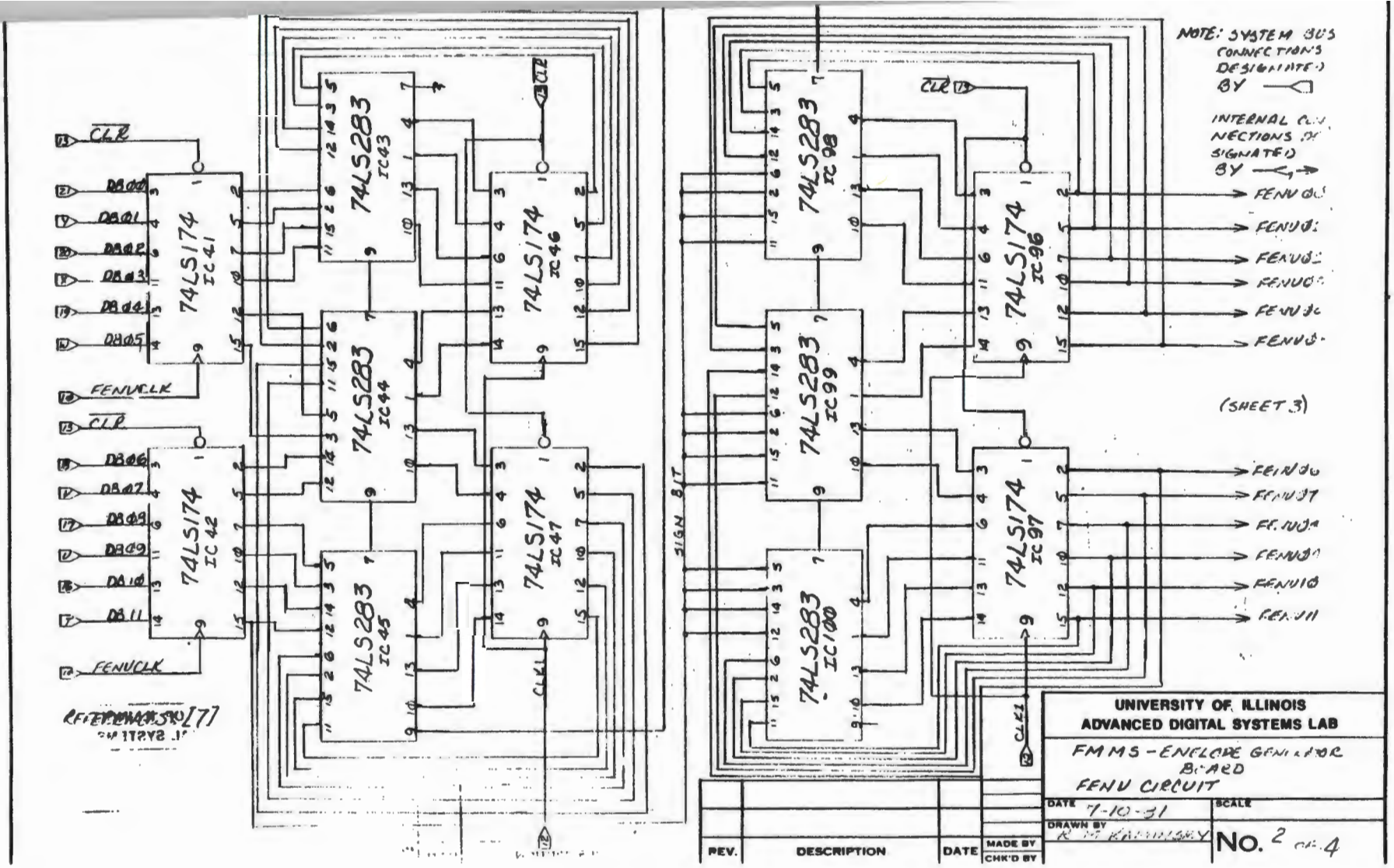


Figure A5 FENV CIRCUIT

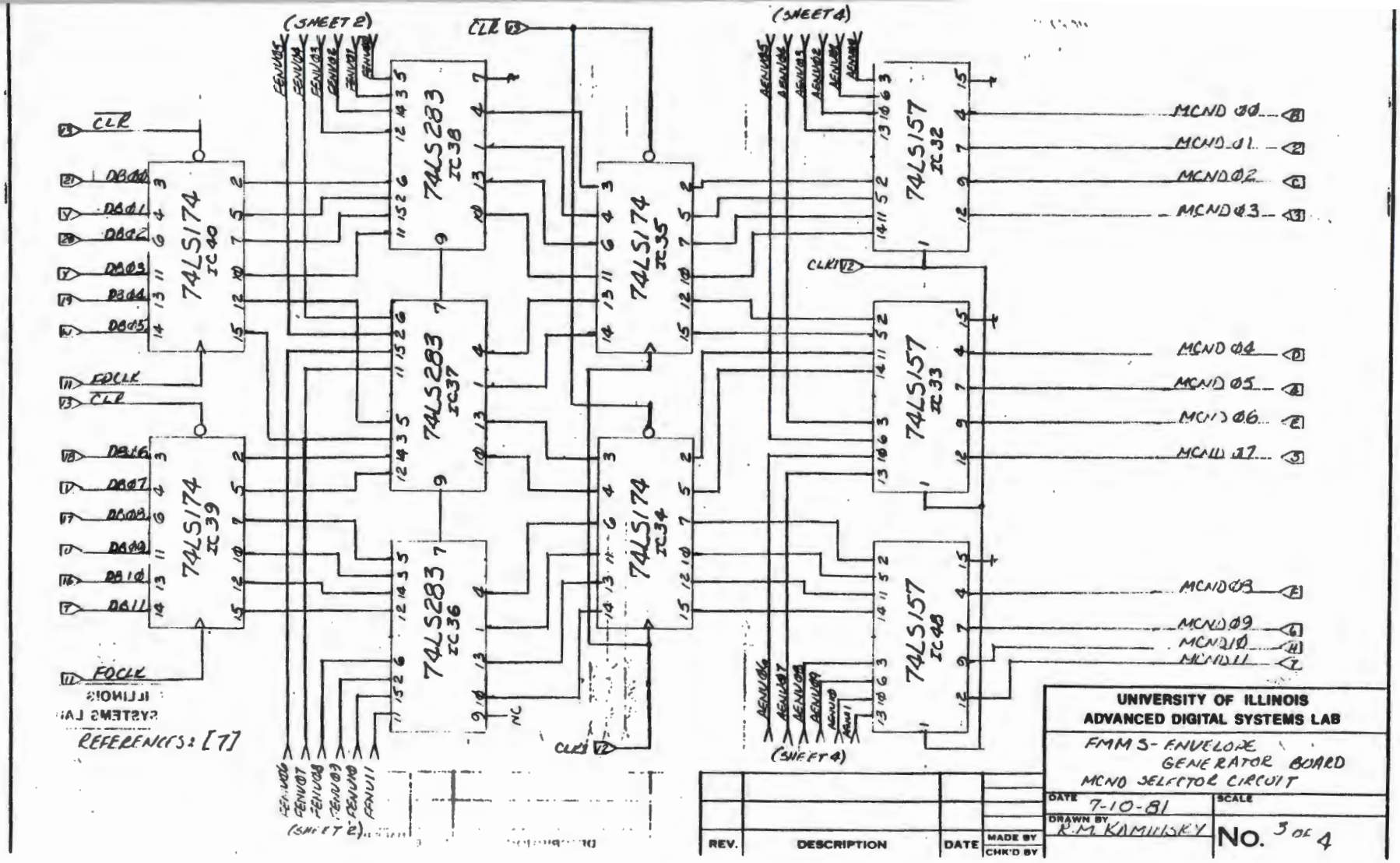


Figure A6 MCND SELECTOR CIRCUIT

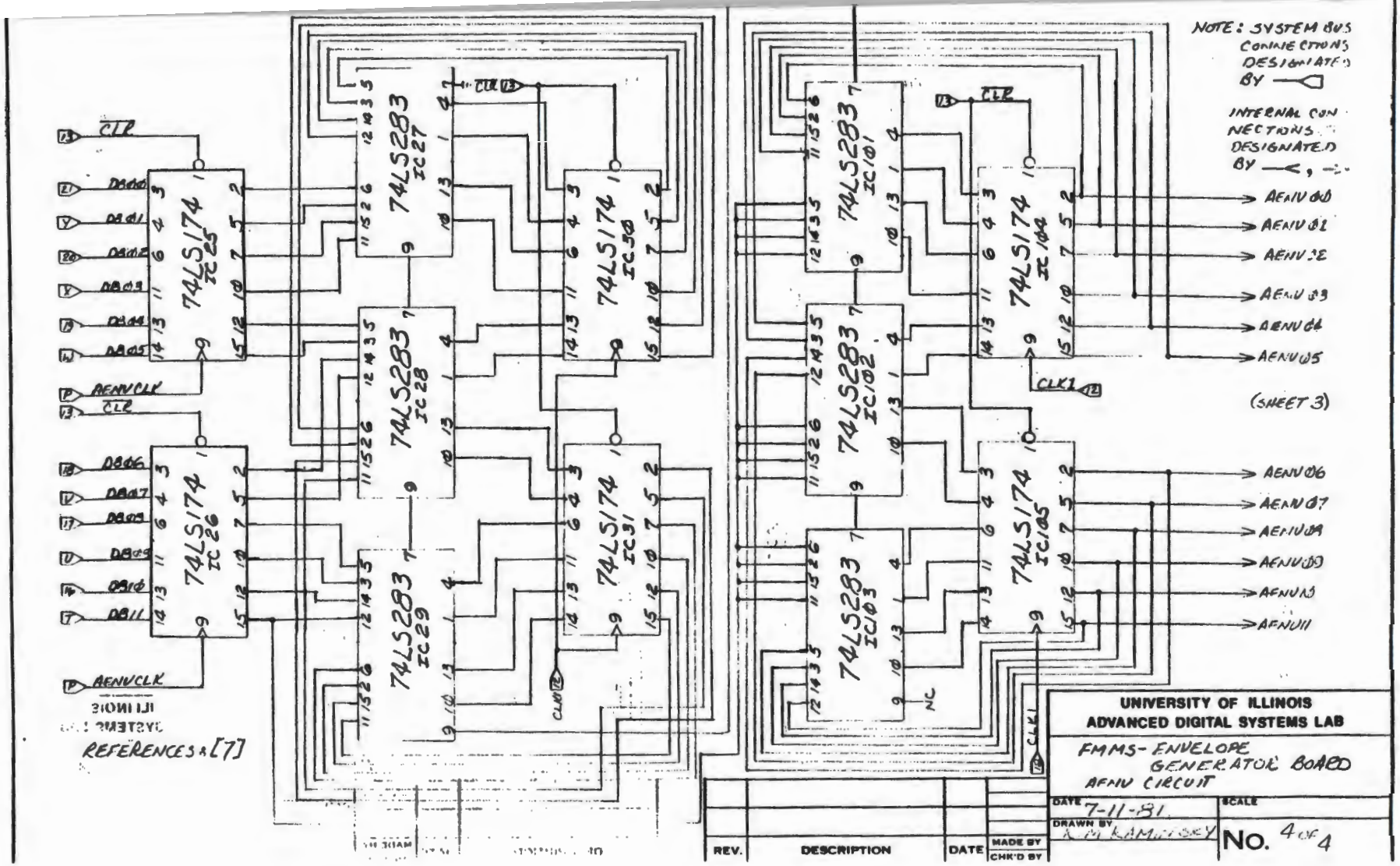
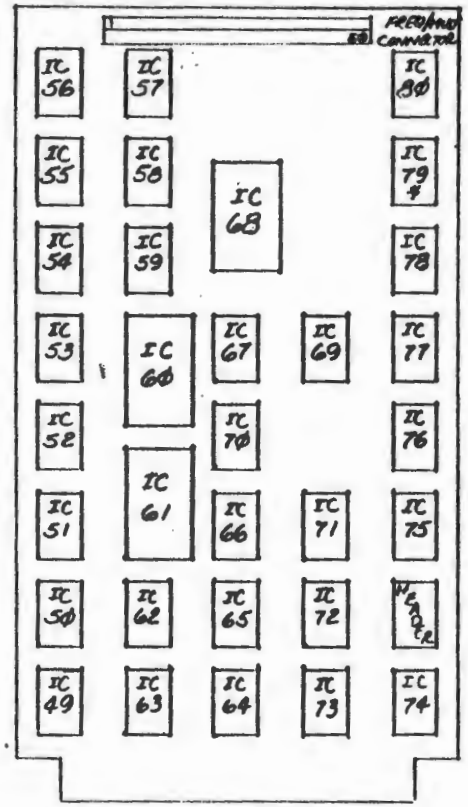


Figure A7 AENV CIRCUIT



WIRING SIDE

& EMPTY SOCKET

PIN	SIGNAL	PIN	SIGNAL
1	+5V	A	GND
2	MCND01	B	MCND00
3	MCND03	C	MCND02
4	MCND05	D	MCND04
5	MCND07	E	MCND06
6	MCND09	F	MCND08
7	MCND11	H	MCND10
8	-15V	J	+15V
9	CLK2	K	OUT
10	NC	L	MCLK
11	NC	M	NC
12	CLK1	N	NC
13	CLR	P	NC
14	NC	R	NC
15	NC	S	NC
16	NC	T	NC
17	NC	U	NC
18	NC	V	NC
19	NC	W	NC
20	NC	X	NC
21	NC	Y	NC
22	GND	Z	+5V

SYSTEM BUS CONNECTOR

PIN	SIGNAL	PIN	SIGNAL
1	MPA03	2	MPA04
3	MPA05	4	MPA06
5	MPA07	6	MPA08
7	MPA09	8	MPA10
9	MPA11	10	MPA12
11	MPA13	12	MPA14
13	MPA15	14	NC
15	CMPA03	16	CMPA04
17	CMPA05	18	CMPA06
19	CMPA07	20	CMPA08
21	CMPA09	22	CMPA10
23	CMPA11	24	CMPA12
25	CMPA13	26	CMPA14
27	CMPA15	28	NC
29	NC	30	NC
31	NC	32	NC
33	NC	34	NC
35	MSWA15	36	MSWA14
37	MSWA13	38	MSWA12
39	MSWA11	40	MSWA10
41	MSWA09	42	MSWA08
43	MSWA07	44	MSWA06
45	MSWA05	46	MSWA04
47	MSWA03	48	MSWA02
49	MSWA01	50	MSWA00

FEED/MULT CONNECTOR

DESCRIPTION OF ACRONYMS

OUT - DAC OUTPUT VOLTAGE

MCLK - MULTIPLIER CLOCK = 25MHz

CLK1 - PIPELINE CLOCK = 2^{15}Hz

CLK2 - TWICE CLK1 = 2^{16}Hz

REV.	DESCRIPTION	DATE	MADE BY / CHK'D BY

UNIVERSITY OF ILLINOIS
ADVANCED DIGITAL SYSTEMS LAB

MULTIPLIER BOARD

COMPONENT LAYOUT & CONNECTOR LIST

DATE 8-9-81

DRAWN BY R. KAMINSKY

SCALE

NO. 1 OF 4

Figure A8 COMPONENT LAYOUT FOR MULTIPLIER BOARD

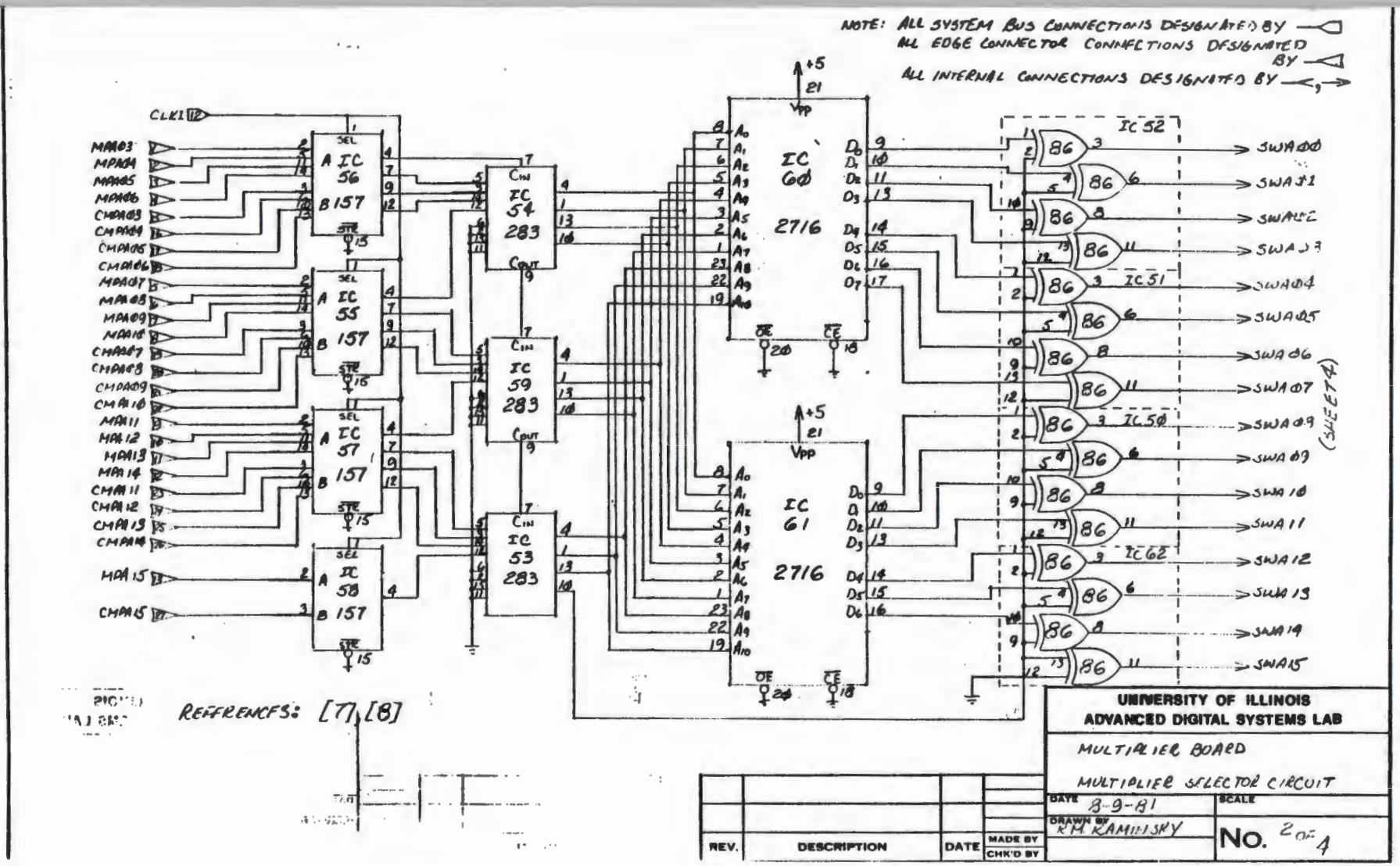


Figure A9 MULTIPLIER SELECTOR CIRCUIT

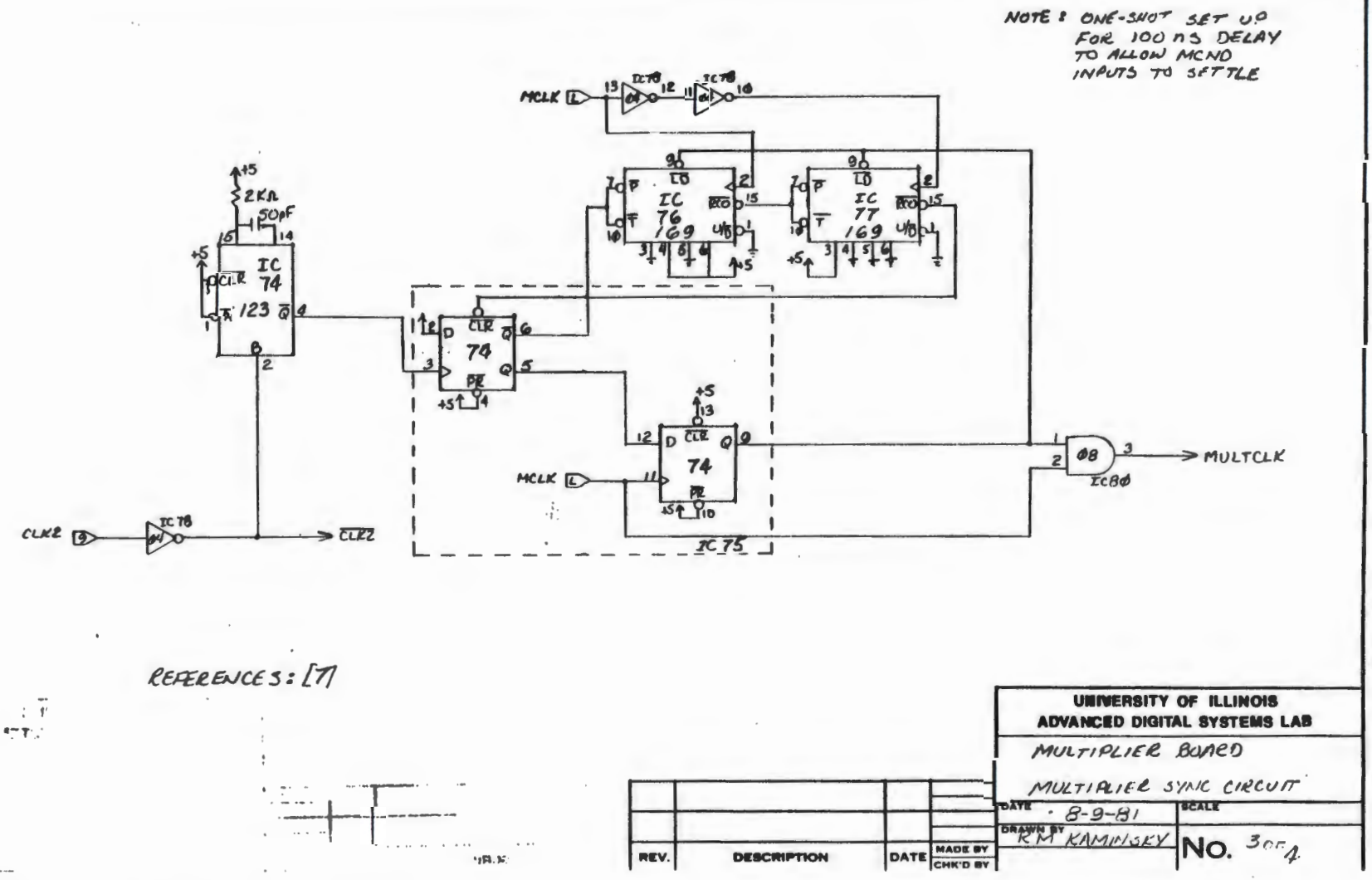


Figure A10 MULTIPLIER SYNCHRONIZATION CIRCUIT

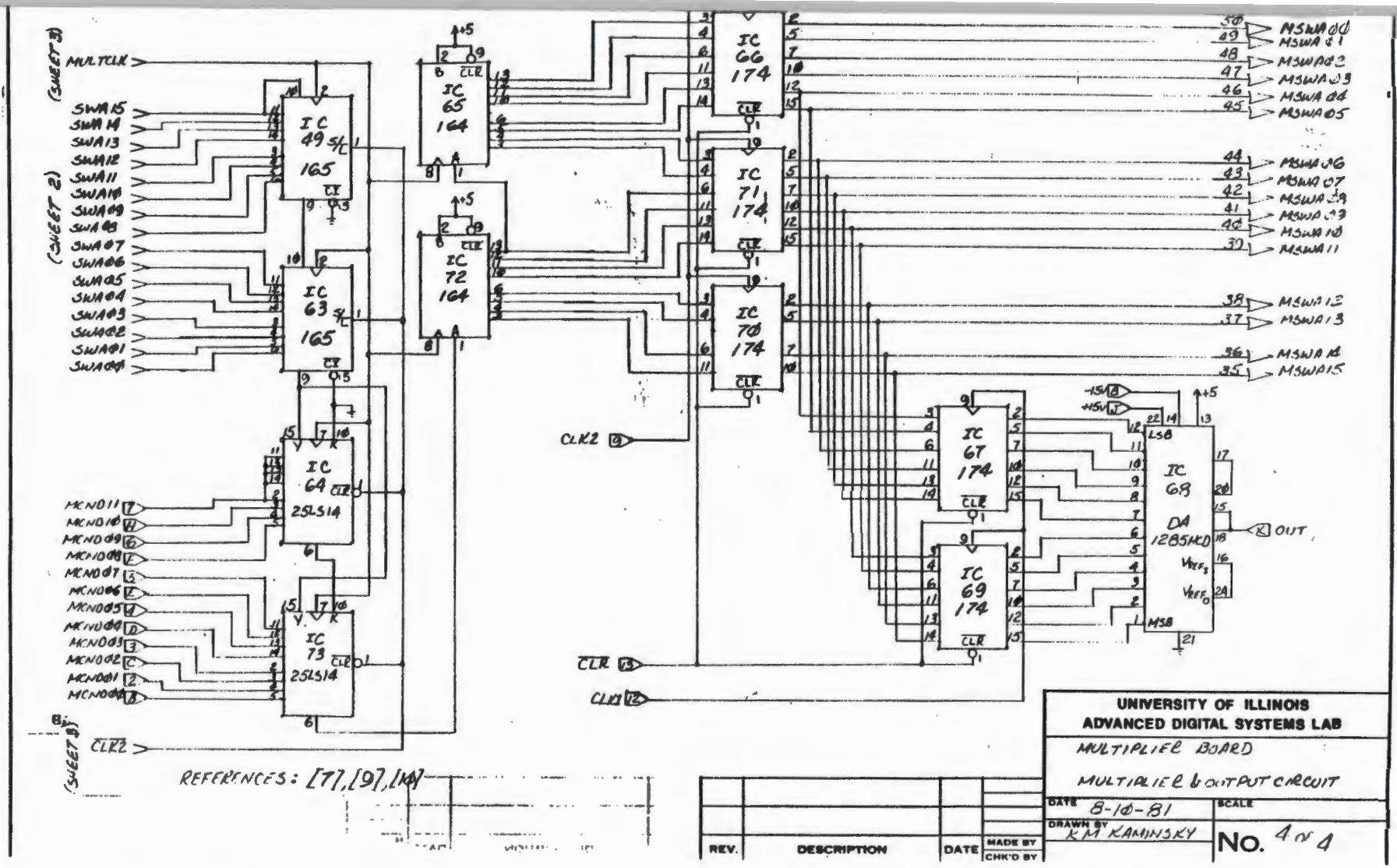
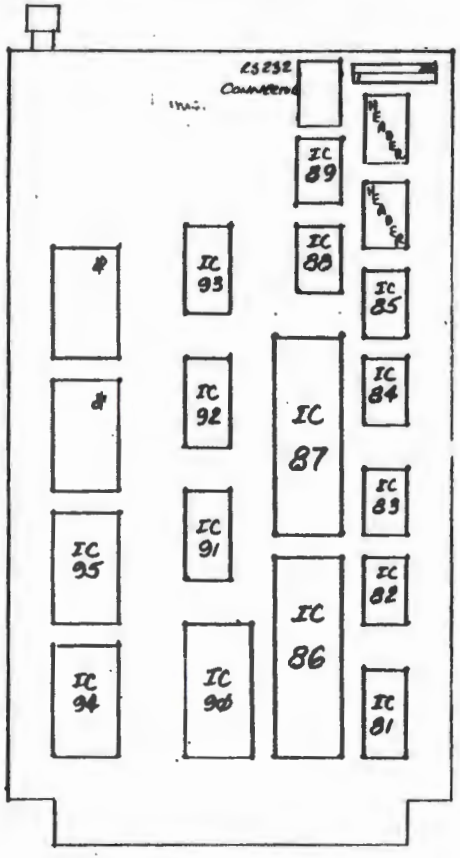


Figure A11 MULTIPLIER & OUTPUT CIRCUIT



PIN	SIGNAL	PIN	SIGNAL
1	+5V	A	GND
2	NC	B	NC
3	NC	C	NC
4	NC	D	NC
5	NC	E	NC
6	NC	F	NC
7	NC	H	NC
8	NC	J	NC
9	CLK2	K	OUT
10	FRMCLK	L	MCLK
11	FDCLK	M	CPUCLK
12	CLK1	N	MMCLK
13	CLR	P	MSYNCLK
14	DB14	R	DB15
15	DB12	S	DB13
16	DB10	T	DB11
17	DB08	U	DB09
18	DB06	V	DB07
19	DB04	W	DB05
20	DB02	X	DB03
21	DB00	Y	DB01
22	GND	Z	+5V

SYSTEM BUS CONNECTOR

PIN	SIGNAL	PIN	SIGNAL
1	GND	2	D0
3	GND	4	D1
5	GND	6	D2
7	GND	8	D3
9	GND	10	D4
11	GND	12	D5
13	GND	14	D6
15	GND	16	D7
17	GND	18	ACK
19	GND	20	$\overline{OBF}$
21	GND	22	NC
23	GND	24	NC
25	GND	26	NC

MUSIC SYSTEM CONNECTOR

REV.	DESCRIPTION	DATE	MADE BY / CHK'D BY

UNIVERSITY OF ILLINOIS
ADVANCED DIGITAL SYSTEMS LAB

PROCESSOR BOARD

COMPONENT LAYOUT & CONNECTOR LIST

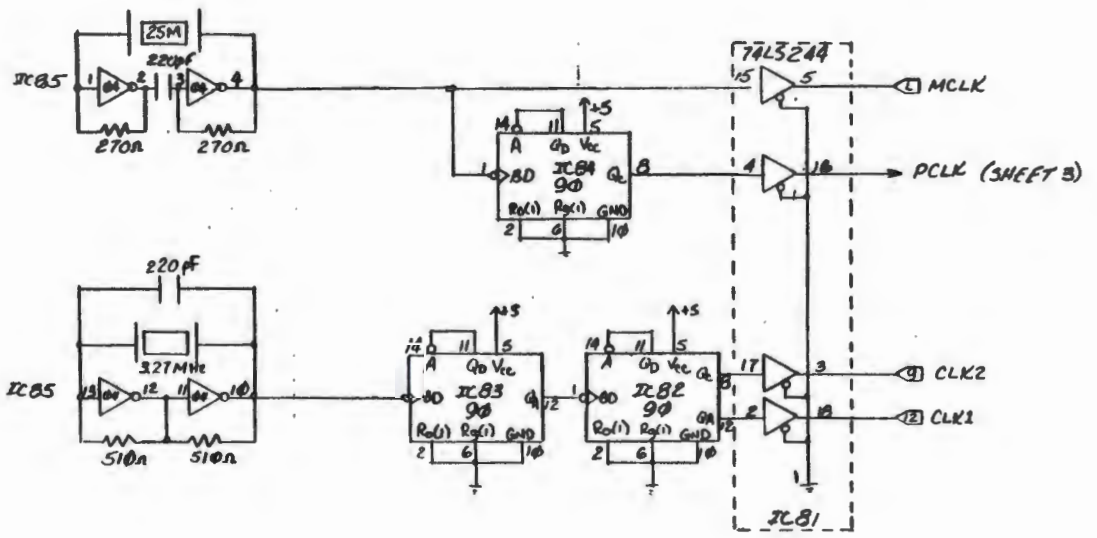
DATE 8-10-81

DRAWN BY K.A. KAMINSKY

SCALE

No. 1 OF 5

Figure A12 COMPONENT LAYOUT FOR PROCESSOR BOARD



REFERENCES: [7]

REV.	DESCRIPTION	DATE	MADE BY / CHK'D BY

UNIVERSITY OF ILLINOIS
ADVANCED DIGITAL SYSTEMS LAB

PROCESSOR BOARD

CLOCK GENERATOR CIRCUIT

DATE 8-10-91 | SCALE

DRAWN BY RM KAMINSKY | NO. 20 of 5

Figure A13 CLOCK GENERATOR CIRCUIT

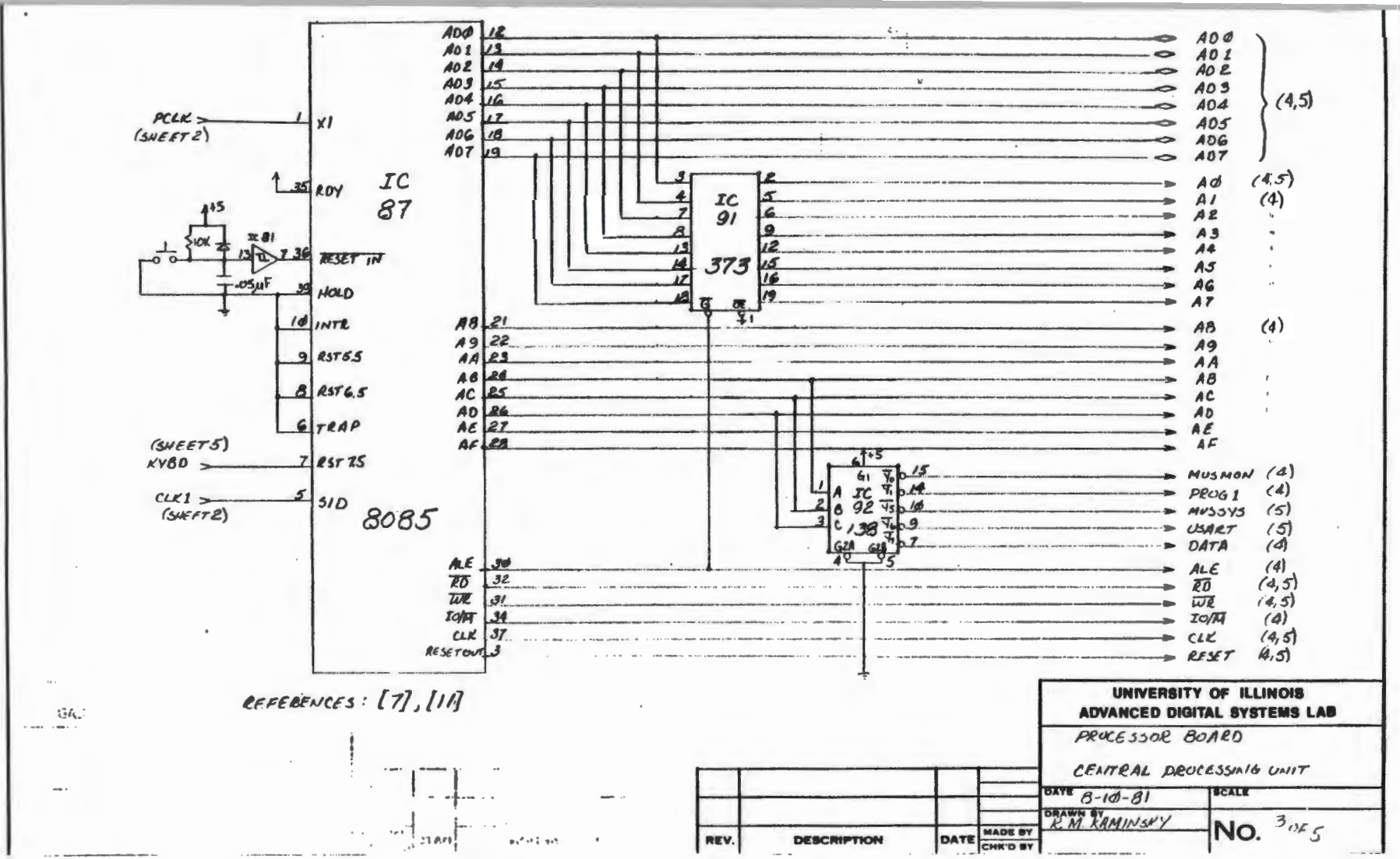
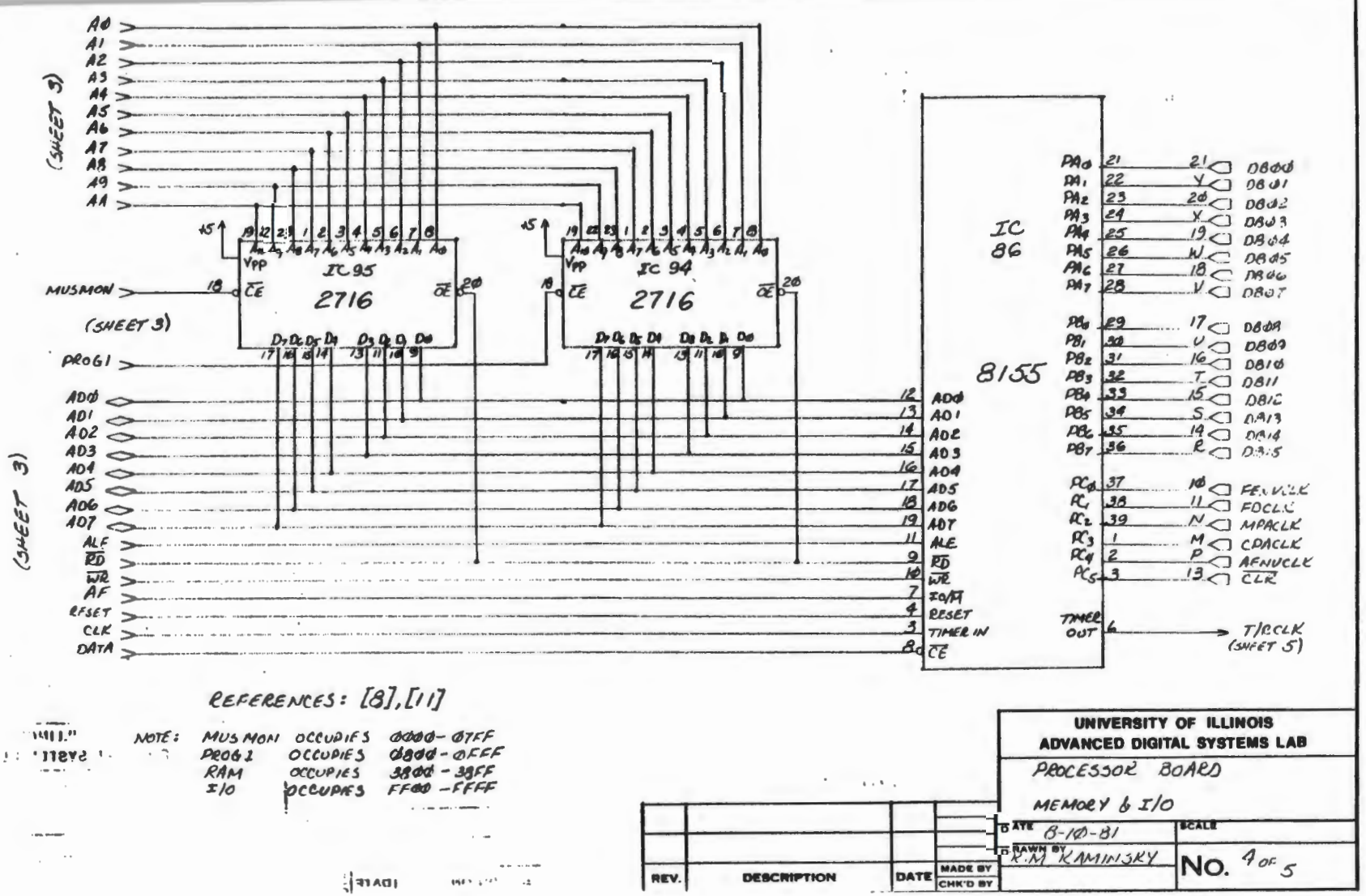


Figure A14 CENTRAL PROCESSING UNIT



REFERENCES: [8], [11]

NOTE: MUSMON OCCUPIES 0000 - 07FF
PROG1 OCCUPIES 0800 - 0FFF
RAM OCCUPIES 3800 - 38FF
I/O OCCUPIES FF00 - FFFF

REV.	DESCRIPTION	DATE	MADE BY / CHK'D BY

UNIVERSITY OF ILLINOIS
ADVANCED DIGITAL SYSTEMS LAB

PROCESSOR BOARD

MEMORY & I/O

DATE 8-10-81

DRAWN BY K.M KAMINSKY

SCALE

NO. 4 OF 5

Figure A15 MEMORY & I/O

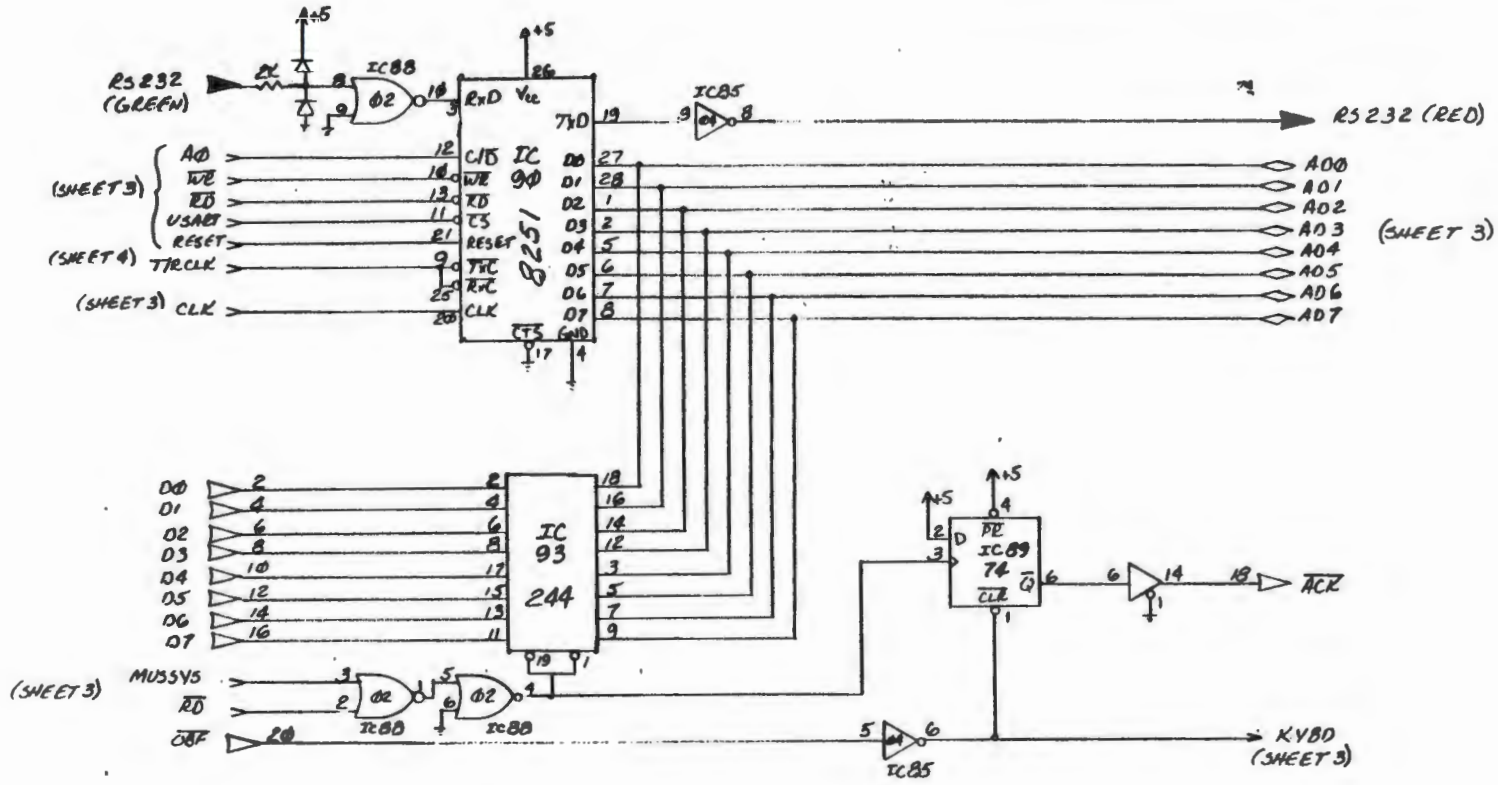


Figure A16 SERIAL I/O & KEYBOARD PORT

APPENDIX B

SYSTEM TIMING DIAGRAM

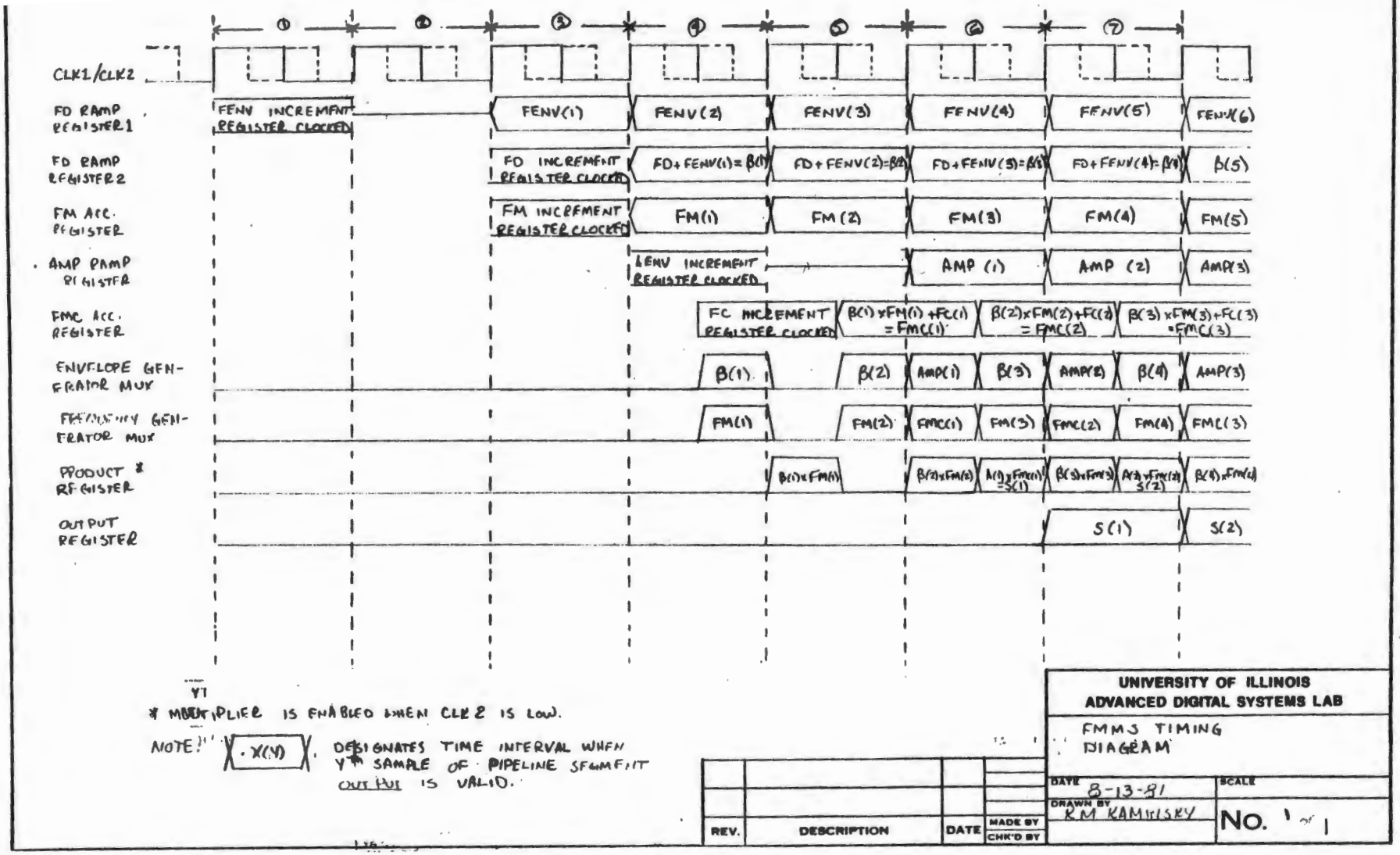


Figure B1 SYSTEM TIMING DIAGRAM

APPENDIX C

TEST PROGRAM SOFTWARE LISTING

```
;*****************************************************
;   THIS IS A TEST PROGRAM TO PLAY A  NOTE WITH      *
;   DYNAMIC CONTROL OF THE MODULATION INDEX AND      *
;   AMPLITUDE ENVELOPES USING THE ADSL FM MUSIC      *
;   SYNTHESIZER.                                     *
;*****************************************************
;
;
;   TYPE THE FOLLOWING DATA INTO RAM FOR TRUMPET-LIKE SOUNDS:
;
;
;    NOTE!    FOR WORDS ENTER LOW BYTE IN LOWER ADDRESS!
;
;
;                        MONITOR INSERTIONS
;
;   LOCATION             DATA          DESCRIPTION
;-------------------------------------------------------
;   3840,41              01EEH           FENV1
;   3842,43              0406H           FM
;   3844,45              0066H           FD
;   3846,47              03F6H           AENV1
;   3848,49              0400H           FC
;   384A                   02H           SD1
;   384B                   02H           SD2
;   384C                   04H           SD3
;   384D                   03H           SD4
;
;   38A0,A1              0000H           FENV2
;   38A2,A3              0000H           FENV3
;   38A4,A5              0F88H           FENV4
;
;   38B0,B1              0EC3H           AENV2
;   38B2,B3              0EC0H           AENV3
;   38B4,B5              0FD0H           AENV4
;
;
```

```
00F9                OPRTA     EQU     0F9H  ;   OUTPUT PORT A
00FA                OPRTB     EQU     0FAH  ;   OUTPUT PORT B
0020                SETCL      EQU      20H  ;    COMMAND TO SET CLEAR PIN
00FB                PORTC      EQU      0FBH  ;    OUTPUT PORT C
FF01                PORTA      EQU    0FF01H  ;    DB LOW BYTE
3800                DATA       EQU     3800H  ;    RAM DATA STARTING ADDRESS
0020                RIM        EQU      20H  ;    RIM OPCODE
3871                TEST       EQU     3871H  ;    BEGINNING ADDRESS OF PROGRAM TEST VALUE
38A0                FENVBUF      EQU     38A0H  ;     STARTING ADDRESS OF FENV BUFFER STORAGE
38AE                FENVPTR      EQU     38AEH  ;     FENV BUFFER POINTER
38B0                AENVBUF      EQU     38B0H  ;     STARTING ADDRESS OF AENV BUFFER STORAGE
38BE                AENVPTR      EQU     38BEH  ;     AENV BUFFER POINTER
                    ;
                    ;
                    ;
0800                          ORG     0800H
0800    F3                     DI
0801    21A038                LXI     H,FENVBUF
0804    22AE38                SHLD    FENVPTR
0807    21B038                LXI     H,AENVBUF
080A    22BE38                SHLD    AENVPTR
080D    AF                    XRA     A
080E    D3FB                  OUT     PORTC  ;    CLEAR PIPELINE REG'S
0810    3E20                  MVI     A,SETCL
0812    D3FB                  OUT     PORTC
                    ;
                    ;
0814    2A4038                LHLD    DATA+64
0817    2201FF                SHLD    PORTA  ;    LOAD FENV1
081A                LOOK       EQU     $
081A    3E69                   MVI     A,69H
081C    327038                 STA     TEST-1  ;     DID PROGRAM GET THIS FAR?
081F    20                     DB      RIM
0820    B7                    ORA     A
0821    F21A08                 JP      LOOK   ;    WAIT FOR CLK1 TO BE HIGH
0824    3E21                  MVI     A,21H
0826    D3FB                  OUT     PORTC  ;    CLOCK-IN FENV1
0828    3E69                  MVI     A,69H
```

```
082A   327138              STA     TEST
                    ;
                    ;
082D   20          LK05:    DB     RIM
082E   B7                  ORA    A
082F   FA2D08                JM     LK05
0832   2A4238               LHLD    DATA+66
0835   2201FF               SHLD    PORTA  ;    LOAD FM
0838                LK10      EQU     $
0838   20                    DB     RIM
0839   B7                   ORA    A
083A   F23808                JP     LK10;    WAIT FOR 1 1/2 PIPELINE CLOCK CYCLES
083D                LK15     EQU    $
083D   20                    DB     RIM
083E   B7                   ORA    A
083F   FA3D08                JM     LK15
0842   3E69                 MVI    A,69H
0844   327238               STA    TEST+1
0847                LK20      EQU    $
0847   20                    DB     RIM
0848   B7                   ORA    A
0849   F24708                JP     LK20
                    ;
                    ;
084C   3E24                 MVI    A,24H
084E   D3FB                  OUT    PORTC  ;    CLOCK-IN FM
0850   2A4438               LHLD    DATA+68
0853   2201FF               SHLD    PORTA  ;    LOAD FD
0856   3E22                 MVI    A,22H
0858   D3FB                  OUT    PORTC  ;    CLOCK-IN FD
085A   3E69                 MVI    A,69H
085C   327338               STA    TEST+2 ;    TEST IF PROGRAM GOT THIS FAR
                    ;
                    ;
085F   2A4638               LHLD    DATA+70
0862   2201FF               SHLD    PORTA  ;    LOAD AENV1
0865   3E30                 MVI    A,30H
0867   D3FB                  OUT    PORTC  ;    CLOCK-IN AENV1
```

```
                        ;
0869                LK25    EQU  $
0869    20              DB   RIM
086A    B7              ORA  A
086B    FA6908          JM   LK25    ;   WAIT FOR CLK1 LOW
086E    2A4838          LHLD DATA+72
0871    2201FF          SHLD PORTA   ;   LOAD FC
0874    3E28            MVI  A,28H
0876    D3FB            OUT  PORTC   ;   CLOCK-IN FC
0878    3E69            MVI  A,69H
087A    327438          STA  TEST+3 ;    TEST IF PROGRAM GOT THIS FAR
                        ;
                        ;
087D    3A4A38          LDA  DATA+74;    LOAD SD1
0880    4F              MOV   C,A
0881    CDCC08      REP:CALL  WAIT
0884    0D              DCR  C
0885    C28108          JNZ  REP
0888    CDD608          CALL SEG
                        ;
088B    3E69            MVI  A,69H
088D    327538          STA  TEST+4 ;    TEST IF PROGRAM GOT THIS FAR
                        ;
0890    3A4B38          LDA  DATA+75;    LOAD SD2
0893    4F              MOV  C,A
0894    CDCC08      REP05:CALL WAIT
0897    0D              DCR  C
0898    C29408          JNZ  REP05
089B    CDD608          CALL SEG
                        ;
                        ;
089E    3A4C38          LDA  DATA+76;    LOAD SD3
08A1    4F              MOV  C,A
08A2    CDCC08      REP10:CALL WAIT
08A5    0D              DCR  C
08A6    C2A208          JNZ  REP10
08A9    CDD608          CALL SEG
```

```
                        ;
                        ;
08AC    3A4D38          LDA     DATA+77;    LOAD SD4
08AF    4F              MOV     C,A
08B0    CDCC08  REP15:  CALL    WAIT
08B3    0D              DCR     C
08B4    C2B008          JNZ     REP15
08B7    AF               XRA    A
08B8    D3FB             OUT    PORTC   ;   CLEAR PIPELINE REG'S
08BA    CDCC08          CALL    WAIT
08BD    CDCC08          CALL    WAIT
08C0    CDCC08          CALL    WAIT
08C3    CDCC08          CALL    WAIT
08C6    CDCC08          CALL    WAIT;   PAUSE BEFORE REPEATING NOTE
08C9    C30008           JMP    0800H   ;   REPEAT TONE
                        ;
                        ;
08CC    21362D  WAIT:    LXI    H,11574D;   DELAY FOR 1/8 SECOND
08CF    2B      LOOP:    DCX    H
08D0    7C               MOV    A,H
08D1    B7               ORA    A
08D2    C2CF08           JNZ    LOOP
08D5    C9               RET
                        ;
                        ;
08D6    2AAE38  SEG:    LHLD    FENVPTR;    LOAD NEXT FREQ. DEV.
                                            ;   ENVELOPE SEGMENT INCREMENT
08D9    7E               MOV    A,M
08DA    D3F9             OUT    OPRTA
08DC    23               INX    H
08DD    7E               MOV    A,M
08DE    D3FA             OUT    OPRTB
08E0    23               INX    H
08E1    22AE38          SHLD    FENVPTR
                        ;
08E4    3E20             MVI    A,SETCL
08E6    D3FB             OUT    PORTC   ;   LOWER CLOCK (IF SET)
08E8    3E21             MVI    A,21H
08EA    D3FB             OUT    PORTC   ;   CLOCK-IN FENV SEG
```

```
                        ;
                        ;
08EC    2ABE38          LHLD    AENVPTR;    LOAD NEXT AMP. ENVELOPE SEGMENT INCREMENT
08EF    7E              MOV     A,M
08F0    D3F9            OUT     OPRTA
08F2    23              INX     H
08F3    7E              MOV     A,M
08F4    D3FA            OUT     OPRTB
08F6    23              INX     H
08F7    22BE38          SHLD    AENVPTR
08FA    3E30            MVI     A,30H
08FC    D3FB            OUT     PORTC   ;   CLOCK-IN AENV REG


08FE    C9              RET
                        END
NO PROGRAM ERRORS
```

SYMBOL TABLE

* 01

A	0007	AENVB	38B0	AENVP	38BE	B	0000
C	0001	D	0002	DATA	3800	E	0003
FENVB	38A0	FENVP	38AE	H	0004	L	0005
LK05	082D	LK10	0838	LK15	083D	LK20	0847
LK25	0869	LOOK	081A	LOOP	08CF	M	0006
OPRTA	00F9	OPRTB	00FA	PORTA	FF01	PORTC	00FB
PSW	0006	REP	0881	REP05	0894	REP10	08A2
REP15	08B0	RIM	0020	SEG	08D6	SETCL	0020
SP	0006	TEST	3871	WAIT	08CC		

APPENDIX D

FMMS MUSMON MONITOR LISTING

```
                ; LAST MODIFICATION 4:45 PM 7/25/81
                ;  BY BOB KAMINSKY
                ;
0000            KBPROC  EQU     0               ;KEYBOARD PROC. INPUT PORT
0008            KBRST   EQU     8               ; OUTPUT PORT TO RESET KBD PROC
38C0            RSTORG  EQU     38C0H           ;WHERE RESTART JUMP TABLE IS
0800            KEYPROG EQU     0800H           ; 2ND ROM SLOT LOCATION
0010            DELFAC  EQU      10H            ; DEFAULT DELAY FACTOR FOR MONITOR
0030            SIM     EQU     30H             ;SIM OPCODE
0020            RIM     EQU     20H             ;RIM OPCODE

4021            BAUD EQU 33+4000H;  FOR 4800 BAUD SQUARE WAVE
38F0            IOTBL   EQU     38F0H; I/O TABLE
001B            BRCHR   EQU     1BH
383D            BRLOC   EQU     383DH;  BREAK LOCATION
0027            CMD     EQU     27H;  USART COMMAND WORD
0031            CNCTL   EQU     031H;  CONSOLE CONTROL
0030            CNIN    EQU     030H;  CONSOLE IN
0030            CNOUT   EQU     030H;  CONSOLE OUT
0031            CONST   EQU     031H
00F8            CTCTL   EQU     0F8H;  COUNTER/TIMER CONTROL
00CF            CTCMD   EQU     0CFH;  COUNTER/TIMER COMMAND
000D            CR      EQU     0DH
3800            DATA    EQU     3800H
001B            ESC     EQU     1BH
000F            HCHAR   EQU     0FH
00FF            INVRT   EQU     0FFH
000A            LF      EQU     0AH
00CE            MODE    EQU     0CEH
0007            NEWLN   EQU     07H
007F            PRTYO   EQU     7FH
382E            REGS    EQU     DATA+64-18
0002            RBR     EQU     2
0038            RSTU    EQU     38H
001B            TERM    EQU     1BH
0001            TRDY    EQU     1
00FF            UPPER   EQU     0FFH
0004            TXBE    EQU     04H
00FC            CTL0    EQU     0FCH;  LOW ORDER 8 DIVISION
00FD            CTHI    EQU     0FDH;  UPPER 2 BITS=TIMER MODE
                                ;  LOWER 6 BITS=HIGH DIVISION RATE
```

```
                ;*****************************************************************************
                ;       COLD START                                              SYSTEM      *
                ;*****************************************************************************
0000                    ORG     0000H
0000    F3              DI
0001    C35100          JMP     STUP
0008                    ORG     8
0008    C3C038          JMP     RS1
0010                    ORG     10H
0010    C3C338          JMP     RS2


0018                    ORG     18H
0018    C3C638          JMP     RS3
0020                    ORG     20H
0020    C3C938          JMP     RS4
0024                    ORG     24H
0024    C3DE38          JMP     TRAP
0028                    ORG     28H
0028    C3CC38          JMP     RS5
002C                    ORG     2CH
002C    C3CF38          JMP     RS55
0030                    ORG     30H
0030    C3D238          JMP     RS6
0034                    ORG     34H
0034    C3D538          JMP     RS65
0038                    ORG     38H
0038    C3D838          JMP     RS7
003C                    ORG     3CH
003C    C3DB38          JMP     RS75

                ;
                ; JUMP TABLE FOR ACCESSING MONITOR ROUTINES
                ;
003F    C38C00          JMP     WARM    ; WARM START ENTRY POINT
0042    C39302          JMP     CO      ; CHAR OUT, W/ DELAY
0045    C3D602          JMP     GETCH   ; READ IN CHAR
0048    C38703          JMP     NMOUT   ; PRINT HEX NUMBER
004B    C3DD02          JMP     GETHX   ; READ HEX NUMBER
004E    C39E02          JMP     PRMSG   ; PRINT STRING
```

```
0051   3E37      STUP:  MVI     A,37H    ;DUMMY CMD WD TO UART TO SET UP SOFT RESET
0053   D331             OUT     CNCTL
0055   3E40             MVI     A,40H    ;SOFTWARE RESET USART
0057   D331             OUT     CNCTL
0059   3ECE             MVI     A,MODE
005B   D331             OUT     CNCTL    ;OUTPUT MODE TO USART
                        ;
                        ; SET UP RESTART JUMP TABLE IN RAM
                        ; TO GO TO THE ROM BREAKPOINT
                        ; ROUTINE
                        ;
005D   160A             MVI     D,10             ;COUNTER
005F   21C038           LXI     H,RSTORG         ;POINT TO RAM RST TABLE
0062   017600           LXI     B,GO
0065   36C3     STLP:  MVI     M,0C3H           ;JUMP OPCODE
0067   23               INX     H
0068   71               MOV     M,C              ;ADDRESS OF "GO"
0069   23               INX     H
006A   70               MOV     M,B
006B   23               INX     H
006C   15               DCR     D                ;CHECK COUNTER


006D   C26500           JNZ     STLP             ;REPEAT LOOP

0070   3EDF             MVI     A,11011111B      ;DISABLE INTERUPT LEVELS
0072   30               DB      SIM              ;SIM INSTRUCTION
0073   C36803           JMP     INUST    ;BRANCH TO COMPLETE INIT
                        ;
0076   223438   GO:    SHLD    LSAVE
0079   E1               POP     H
007A   223638           SHLD    PSAVE
007D   F5               PUSH    PSW
007E   210200           LXI     H,2
0081   39               DAD     SP
0082   223838           SHLD    SSAVE
0085   F1               POP     PSW
0086   313438           LXI     SP,ASAVE+1
0089   C35002           JMP     ADROUT
```

```
                        ;*****************************************************************************
                        ;       PRINT SIGNON MESSAGE                                    SYSTEM      *
                        ;*****************************************************************************
008C                    WARM:                   ; WARM START ENTRY POINT
008C    215F04          SGMSG:  LXI     H,SGNON
008F    CD9E02                  CALL    PRMSG
0092    C39500                  JMP     GETCM
                        ;*************************************************
                        ;       COMMAND RECOGNIZING ROUTINE
                        ;*************************************************
0095    312E38          GETCM:  LXI     SP,MSTAK
0098    0E3E                    MVI     C,'>'
009A    CDAF02                  CALL    ECHO
                        ;
009D    CDD602          GTC03:  CALL    GETCH
00A0    CDAF02                  CALL    ECHO
00A3    79                      MOV     A,C
00A4    010A00                  LXI     B,NCMDS
00A7    219304                  LXI     H,CTAB
00AA    BE              GTC05:  CMP     M
00AB    CAB600                  JZ      GTC10
00AE    23                      INX     H
00AF    0D                      DCR     C
00B0    C2AA00                  JNZ     GTC05
00B3    C3C802                  JMP     ERROR
00B6    217D04          GTC10:  LXI     H,CADR
00B9    09                      DAD     B
00BA    09                      DAD     B
00BB    7E                      MOV     A,M
00BC    23                      INX     H
00BD    66                      MOV     H,M
00BE    6F                      MOV     L,A
00BF    E9                      PCHL
                        ;
00C0    0E02            DCMD:   MVI     C,2



00C2    CD1103                  CALL    GETNM
00C5    D1                      POP     D
00C6    E1                      POP     H
00C7    CDAA02          DCM05:  CALL    CROUT
00CA    CD4702                  CALL    ADRD
```

```
00CD   0E20      DCM10:  MVI     C,' '
00CF   CDAF02            CALL    ECHO
00D2   7E                MOV     A,M
00D3   CD8703            CALL    NMOUT
00D6   CD6102            CALL    BREAK
00D9   DACD02            JC      EXIT
00DC   CD5603            CALL    HILO
00DF   DACD02            JC      EXIT
00E2   23                INX     H
00E3   7D                MOV     A,L
00E4   E60F              ANI     0FH
00E6   C2CD00            JNZ     DCM10
00E9   C3C700            JMP     DCM05
                ;
00EC   CDDD02    GCMD:   CALL    GETHX
00EF   D20101            JNC     GCM05
00F2   7A                MOV     A,D
00F3   FE0D              CPI     CR
00F5   C2C802            JNZ     ERROR
00F8   213638            LXI     H,PSAVE
00FB   71                MOV     M,C
00FC   23                INX     H
00FD   70                MOV     M,B
00FE   C30701            JMP     GCM10
0101   7A        GCM05:  MOV     A,D
0102   FE0D              CPI     CR
0104   C2C802            JNZ     ERROR
0107   C3EC03    GCM10:  JMP     RSTTF
                ;
                ;
010A   21CC05    CCMD:   LXI     H,CMSG
010D   CD9E02            CALL    PRMSG
0110   C39500            JMP     GETCM
```

```
                        ;
                        ; NEW FORMAT FOR OUTPUT COMMAND:
                        ; O <PORT>,<DATA>
0113    CDDD02  OCMD:   CALL    GETHX
0116    D2C802          JNC     ERROR
0119    7A              MOV     A,D
011A    FE20            CPI     ' '
011C    CA2401          JZ      OC05
011F    FE2C            CPI     ','
0121    C2C802          JNZ     ERROR
0124    C5      OC05:   PUSH    B   ;  PUT PORT # ON STACK
0125    CDDD02          CALL    GETHX
0128    D2C802          JNC     ERROR



012B    7A              MOV     A,D
012C    FE0D            CPI     CR
012E    C2C802          JNZ     ERROR
0131    C5              PUSH    B    ; PUT DATA ON STACK
0132    CD3902          CALL    SETUP
0135    C1              POP     B
0136    D1              POP     D
0137    7B              MOV     A,E
0138    32F138          STA     IOTBL+1
013B    79              MOV     A,C
013C    CDF038          CALL    IOTBL
013F    C39500          JMP     GETCM
                        ;
0142    CDDD02  ICMD:   CALL    GETHX
0145    D2C802          JNC     ERROR
0148    7A              MOV     A,D
0149    FE3D            CPI     '='
014B    CA5301          JZ      ICM01
014E    FE0D            CPI     CR
0150    C2C802          JNZ     ERROR
0153    C5      ICM01:  PUSH    B
0154    CD3902          CALL    SETUP
0157    C1              POP     B
0158    79              MOV     A,C
0159    32F438          STA     IOTBL+4
015C    CDF338          CALL    IOTBL+3
015F    4F              MOV     C,A
0160    CD8703          CALL    NMOUT
0163    CDAA02          CALL    CROUT
0166    C39500          JMP     GETCM
```

```
                        ;
0169    0E03    MCMD:   MVI     C,3
016B    CD1103          CALL    GETNM
016E    C1              POP     B
016F    E1              POP     H
0170    D1              POP     D
0171    CD7701          CALL    MCM05
0174    C39500          JMP     GETCM
0177    E5      MCM05:  PUSH    H
0178    62              MOV     H,D
0179    6B              MOV     L,E
017A    7E              MOV     A,M
017B    60              MOV     H,B
017C    69              MOV     L,C
017D    77              MOV     M,A
017E    03              INX     B
017F    78              MOV     A,B
0180    B1              ORA     C
0181    C8              RZ
0182    13              INX     D
0183    E1              POP     H


0184    CD5603          CALL    HILO
0187    D0              RNC
0188    C37701          JMP     MCM05
                        ;
018B    0E01    SCMD:   MVI     C,1
018D    CD1103          CALL    GETNM
0190    E1              POP     H
0191    C39F01          JMP     SCM10
0194    7A      SCM05:  MOV     A,D
0195    FE20            CPI     ' '
0197    CA9F01          JZ      SCM10
019A    FE2C            CPI     ','
019C    C29500          JNZ     GETCM
019F    7E      SCM10:  MOV     A,M
01A0    CD8703          CALL    NMOUT
01A3    0E2D            MVI     C,'-'
01A5    CDAF02          CALL    ECHO
01A8    CDDD02          CALL    GETHX
01AB    D2AF01          JNC     SCM15
01AE    71              MOV     M,C
```

```
01AF   23       SCM15:  INX     H
01B0   7D               MOV     A,L
01B1   E60F             ANI     0FH
01B3   C29401           JNZ     SCM05
01B6   CDAA02           CALL    CROUT
01B9   CD4702           CALL    ADRD
01BC   0E20             MVI     C,' '
01BE   CDAF02           CALL    ECHO
01C1   C39401           JMP     SCM05
                ;
01C4   CDD602   XCMD:   CALL    GETCH
01C7   CDAF02           CALL    ECHO
01CA   79               MOV     A,C
01CB   FE0D             CPI     CR
01CD   C2D601           JNZ     XCM05
01D0   CDA403           CALL    REGDS
01D3   C39500           JMP     GETCM
01D6   4F       XCM05:  MOV     C,A
01D7   CDD503           CALL    RGADR
01DA   C5               PUSH    B
01DB   E1               POP     H
01DC   0E20             MVI     C,' '
01DE   CDAF02           CALL    ECHO
01E1   79               MOV     A,C
01E2   323A38           STA     TEMP
01E5   3A3A38   XCM10:  LDA     TEMP
01E8   FE20             CPI     ' '
01EA   CAF201           JZ      XCM15
01ED   FE2C             CPI     ','
01EF   C29500           JNZ     GETCM
```

```
01F2   7E        XCM15:  MOV     A,M


01F3   B7                MORA    A
01F4   CACD02            JZ      EXIT
01F7   E5                PUSH    H
01F8   5E                MOV     E,M
01F9   1638              MVI     D,DATA/256
01FB   23                INX     H
01FC   46                MOV     B,M
01FD   D5                PUSH    D
01FE   D5                PUSH    D
01FF   E1                POP     H
0200   C5                PUSH    B
0201   7E                MOV     A,M
0202   CD8703            CALL    NMOUT
0205   F1                POP     PSW
0206   F5                PUSH    PSW
0207   B7                MORA    A
0208   CA1002            JZ      XCM20
020B   2B                DCX     H
020C   7E                MOV     A,M
020D   CD8703            CALL    NMOUT
0210   0E2D      XCM20:  MVI     C,'-'
0212   CDAF02            CALL    ECHO
0215   CDDD02            CALL    GETHX
0218   D23002            JNC     XCM30
021B   7A                MOV     A,D
021C   323A38            STA     TEMP
021F   F1                POP     PSW
0220   E1                POP     H
0221   B7                MORA    A
0222   CA2702            JZ      XCM25
0225   70                MOV     M,B
0226   2B                DCX     H
```

```
0227   71       XCM25:  MOV     M,C
0228   110300   XCM27:  LXI     D,RTABS
022B   E1               POP     H
022C   19               DAD     D
022D   C3E501           JMP     XCM10
0230   7A       XCM30:  MOV     A,D
0231   323A38           STA     TEMP
0234   D1               POP     D
0235   D1               POP     D
0236   C32802           JMP     XCM27
                ;
0239   01F038   SETUP:  LXI     B,IOTBL
023C   118606           LXI     D,TBL1
023F   210500           LXI     H,5
0242   19               DAD     D
0243   CD7701           CALL    MCM05
0246   C9               RET
                ;*****************************************************************************
                ;       ADRD                                                    UTILITY     *


                ;       INPUTS: HL-ADDRESS TO BE DISPLAYED                                  *
                ;       CALLS: NMOUT                                                        *
                ;       DESTROYS: A                                                         *
                ;       FUNCTION: OUTPUTS TO THE CONSOLE THE ADDRESS CONTAINED IN THE HL    *
                ;                 REGISTER PAIR                                             *
                ;*****************************************************************************
0247   7C       ADRD:   MOV     A,H
0248   CD8703           CALL    NMOUT
024B   7D               MOV     A,L
024C   CD8703           CALL    NMOUT
024F   C9               RET
                ;
0250   F5       ADROUT: PUSH    PSW
0251   C5               PUSH    B
0252   D5               PUSH    D
0253   0E23             MVI     C,'#'
0255   CDAF02           CALL    ECHO
0258   2A3638           LHLD    PSAVE
025B   CD4702           CALL    ADRD
025E   C3CD02           JMP     EXIT
```

```
                        ;*****************************************************************************
                        ;       BREAK                                                   UTILITY     *
                        ;       OUTPUTS: CARRY=1 IF ESCAPE CHARACTER INPUT                          *
                        ;                =0 IF ANY OTHER CHARACTER OR NO CHARACTER                  *
                        ;       DESTROYS: A,F/F'S                                                   *
                        ;       FUNCTION: BREAK IS USED TO SENSE AN ESCAPE CHARACTER FROM THE USER. *
                        ;                 IF NO CHARACTER IS PENDING, OR IF THE CHARACTER IS NOT    *
                        ;                 AN ESCAPE, THEN A FAILURE RETURN (CARRY=0) IS TAKEN.      *
                        ;                 IN THIS CASE, THE CHARACTER IS LOST.  IF THE CHARACTER    *
                        ;                 IS AN ESCAPE, THEN RETURN IS SUCCESSFUL (CARRY=1).        *
                        ;*****************************************************************************
0261    DB31    BREAK:  IN      CONST           ;GET URART STATUS
0263    E602            ANI     RBR
0265    CAD302          JZ      FRET            ;RETURN FAILURE IF NO CHAR
0268    DB30    BR05:   IN      CNIN            ;ELSE, GET CHAR
026A    E67F            ANI     PRTY0
026C    FE1B            CPI     BRCHR
026E    CA0004          JZ      SRET            ;IF ESCAPE, THEN SUCCSEFULL RETURN
0271    FE13            CPI     13H
0273    C2D302          JNZ     FRET            ;IF NOT <CTR>S, THEN FAIL RETURN
0276    DB31    BR10:   IN      CONST           ;ELSE, WAIT FOR NEW CHAR
0278    E602            ANI     RBR
027A    CA7602          JZ      BR10
027D    C36802          JMP     BR05
                        ;*****************************************************************************
                        ;       CI                                                      UTILITY     *
                        ;       OUTPUTS: A-CHARACTER FROM CONSOLE                                   *
                        ;       DESTROYS: A,F/F'S                                                   *
                        ;       FUNCTION: CI WAITS UNTIL A CHARACTER HAS BEEN ENTERED AT THE CONSOLE *
                        ;                 AND THEN RETURNS THE CHARACTER, VIA THE A REGISTER, TO THE *
                        ;                 CALLING ROUTINE.                                          *
                        ;*****************************************************************************
0280    DB31    CI:     IN      CONST
0282    E602            ANI     RBR
0284    CA8002          JZ      CI
0287    DB30            IN      CNIN
0289    C9              RET
```

```
                        ;*****************************************************************************
                        ;       CNVBN                                                   UTILITY     *
                        ;       INPUTS: C-ASCII CHARACTER '0'-'9' OR 'A'-'F'                        *
                        ;       OUTPUTS: A- 0 TO F HEX                                              *
                        ;       DESTROYS: A,F/F'S                                                   *
                        ;       FUNCTION: CNVBN CONVERTS THE ASCII REPRESENTATION OF A HEX CHARACTER *
                        ;                 INTO ITS CORRESPONDING BINARY VALUE.  CNVBN DOES NOT CHECK *
                        ;                 THE VALIDITY OF ITS INPUT.                                *
                        ;*****************************************************************************
028A    79      CVNBN:  MOV     A,C
028B    D630            SUI     '0'
028D    FE0A            CPI     10D
028F    F8              RM
0290    D607            SUI     7
0292    C9              RET
                        ;*****************************************************************************
                        ;       CO                                                      UTILITY     *
                        ;       INPUTS: C-CHARACTER TO OUTPUT TO CONSOLE                            *
                        ;       OUTPUTS: C-CHARACTER OUTPUT TO CONSOLE                              *
                        ;       DESTROYS: A,F/F'S                                                   *
                        ;       FUNCTION: CO WAITS UNTIL THE CONSOLE IS READY TO ACCEPT A CHARACTER *
                        ;                 AND THEN SENDS THE INPUT ARGUMENT TO THE CONSOLE.         *
                        ;*****************************************************************************
0293    DB31    CO:     IN      CONST
0295    E601            ANI     TRDY
0297    CA9302          JZ      CO
029A    79              MOV     A,C
029B    D330            OUT     CNOUT
029D    C9              RET


029E    7E      PRMSG:  MOV     A,M
029F    FEFF            CPI     OFFH    ; END OF STRING?
02A1    C8              RZ
02A2    4E              MOV     C,M
02A3    CD9302          CALL    CO
02A6    23              INX     H
02A7    C39E02          JMP     PRMSG
```

```
                ;*****************************************************************************
                ;       CROUT                                                   UTILITY     *
                ;       CALLS: ECHO                                                         *
                ;       DESTROYS: A,B,C,F/F'S                                               *
                ;       FUNCTION: CROUT SENDS A CARRIAGE RETURN (AND HENCE A LINE FEED) TO  *


                ;                 THE CONSOLE DEVICE.                                       *
                ;*****************************************************************************
02AA    0E0D    CROUT:  MVI     C,CR
02AC    C3AF02          JMP     ECHO
                ;*****************************************************************************
                ;       ECHO                                                    UTILITY     *
                ;       INPUTS: C-CHARACTER TO ECHO TO CONSOLE                              *
                ;       OUTPUTS: C-CHARACTER ECHOED TO CONSOLE                              *
                ;       CALLS: CO                                                           *
                ;       DESTROYS: A,B,F/F'S                                                 *
                ;       FUNCTION: ECHO TAKES A SINGLE CHARACTER AS INPUT AND, VIA THE       *
                ;                 MONITOR, SENDS THAT CHARACTER TO THE USER CONSOLE.         *
                ;                 A CARRIAGE RETURN IS ECHOED AS A CARRIAGE RETURN LINE     *
                ;                 FEED, AND AN ESCAPE IS ECHOED AS '$'.                     *
                ;*****************************************************************************
02AF    41      ECHO:   MOV     B,C
02B0    3E1B            MVI     A,ESC
02B2    B8              CMP     B
02B3    C2B802          JNZ     ECH05
02B6    0E24            MVI     C,'$'
02B8    CD9302  ECH05:  CALL    CO
02BB    3E0D            MVI     A,CR
02BD    B8              CMP     B
02BE    C2C602          JNZ     ECH10
02C1    0E0A            MVI     C,LF
02C3    CD9302          CALL    CO
02C6    48      ECH10:  MOV     C,B
02C7    C9              RET
```

```
                        ;*****************************************************************************
                        ;       ERROR                                                   UTILITY     *
                        ;       CALLS: ECHO,CROUT,GETCM                                             *
                        ;       DESTROYS: A,B,C,F/F'S                                               *
                        ;       FUNCTION: ERROR PRINTS THE AN ASTERISK ON THE CONSOLE, FOLLOWED BY  *
                        ;                 A CARRIAGE RETURN LINE FEED, AND THEN RETURNS CONTROL     *
                        ;                 TO THE COMMAND RECOGNIZER.                                *
                        ;*****************************************************************************
02C8    0E23    ERROR:  MVI     C,'#'
02CA    CDAF02          CALL    ECHO
02CD    CDAA02  EXIT:   CALL    CROUT
02D0    C39500          JMP     GETCM
                        ;*****************************************************************************
                        ;       FRET                                                    UTILITY     *
                        ;       DESTROYS: CARRY                                                     *
                        ;       FUNCTION: FRET SETS THE CARRY FALSE, DENOTING FAILURE, AND THEN     *
                        ;                 RETURNS TO THE CALLER OF THE ROUTINE INVOKING FRET.       *
                        ;*****************************************************************************
02D3    37      FRET:   STC
02D4    3F              CMC
02D5    C9              RET
                        ;*****************************************************************************
                        ;       GETCH                                                   UTILITY     *


                        ;       OUTPUTS: C-NEXT CHARACTER IN INPUT STREAM                           *
                        ;       CALLS: CI                                                           *
                        ;       DESTROYS: A,C,F/F'S                                                 *
                        ;       FUNCTION: GETCH RETURNS THE NEXT CHARACTER IN THE INPUT STREAM TO    *
                        ;                 THE CALLING ROUTINE.                                      *
                        ;*****************************************************************************
02D6    CD8002  GETCH:  CALL    CI
02D9    E67F            ANI     PRTYO
02DB    4F              MOV     C,A
02DC    C9              RET
```

```
;*****************************************************************************
;       GETHEX                                                  UTILITY      *
;       OUTPUTS: BC-16 BIT INTEGER                                           *
;                D-CHARACTER WHICH TERMINATED INTEGER                        *
;                CARRY=1 IF FIRST CHARACTER NOT TERMINATOR                   *
;                     =0 IF FIRST CHARACTER IS TERMINATOR                    *
;       CALLS: GETCH,ECHO,VALDL,VALDG,CNVBN,ERROR                            *
;       DESTROYS: A,B,C,D,E,F/F'S                                            *
;       FUNCTION: GETHX ACCEPTS A STRING OF HEX DIGITS FROM THE INPUT        *
;                 STREAM AND RETURNS THEIR VALUE AS A 16-BIT BINARY INTEGER. *
;                 IF MORE THAN 4 HEX DIGITS ARE ENTERED, ONLY THE LAST 4     *
;                 ARE USED.  THE NUMBER TERMINATES WHEN A VALID DELIMITER    *
;                 IS ENCOUNTERED.  THE TERMINATOR IS RETURNED AS AN OUTPUT   *
;                 OF THE FUNCTION.  ILLEGAL CHARACTERS CAUSE AN ERROR IN-    *
;                 DICATION.  IF THE FIRST CHARACTER IN THE INPUT STREAM      *
;                 IS NOT A VALID DELIMETER, GETHX WILL RETURN WITH THE       *
;                 CARRY BIT SET TO 1;  OTHERWISE, THE CARRY BIT IS SET TO 0  *
;                 AND THE CONTENTS OF BC PAIR ARE UNDEFINED.                 *
;*****************************************************************************
02DD    E5          GETHX:  PUSH    H
02DE    210000              LXI     H,0
02E1    1E00                MVI     E,0
02E3    CDD602      GHX05:  CALL    GETCH
02E6    CDAF02              CALL    ECHO
02E9    CD4704              CALL    VALDL
02EC    D2FB02              JNC     GHX10
02EF    51                  MOV     D,C
02F0    E5                  PUSH    H
02F1    C1                  POP     B
02F2    E1                  POP     H
02F3    7B                  MOV     A,E
02F4    B7                  ORA     A
02F5    C20004              JNZ     SRET
02F8    CAD302              JZ      FRET
02FB    CD2C04      GHX10:  CALL    VALDG
02FE    D2C802              JNC     ERROR
0301    CD8A02              CALL    CVNBN
0304    1EFF                MVI     E,INVRT
0306    29                  DAD     H
0307    29                  DAD     H
0308    29                  DAD     H
```

```
0309    29              DAD     H
030A    0600            MVI     B,00
030C    4F              MOV     C,A
030D    09              DAD     B
030E    C3E302          JMP     GHX05
                ;*****************************************************************************
                ;       GETNM                                                   UTILITY     *
                ;       INPUTS: C-COUNT OF NUMBERS TO FIND IN INPUT STREAM                  *
                ;       OUTPUTS: TOP OF STACK-NUMBERS FOUND IN REVERSE ORDER                *
                ;       CALLS: GETHX,HILO,ERROR                                             *
                ;       DESTROYS: A,B,C,D,E,H,L,F/F'S                                       *
                ;       FUNCTION: GETNM FINDS A SPECIFIED COUNT OF NUMBERS, BETWEEN 1 AND 3,*
                ;                 INCLUSIVE, IN THE INPUT STREAM AND RETURNS THEIR VALUES   *
                ;                 ON THE STACK.  IF 2 OR MORE NUMBERS ARE REQUESTED, THEN   *
                ;                 THE FIRST MUST BE LESS THAN OR EQUAL TO THE SECOND, OR    *
                ;                 THE FIRST AND SECOND NUMBERS WILL BE SET EQUAL.  THE LAST *
                ;                 NUMBER REQUESTED MUST BE TERMINATED BY A CARRIAGE RETURN  *
                ;                 OR AN ERROR WILL RESULT.                                  *
                ;*****************************************************************************
0311    2E03    GETNM:  MVI     L,03
0313    79              MOV     A,C
0314    E603            ANI     03
0316    C8              RZ
0317    67              MOV     H,A
0318    CDDD02  GNM05:  CALL    GETHX
031B    D2C802          JNC     ERROR
031E    C5              PUSH    B
031F    2D              DCR     L
0320    25              DCR     H
0321    CA2D03          JZ      GNM10
0324    7A              MOV     A,D
0325    FE0D            CPI     CR
0327    CAC802          JZ      ERROR
032A    C31803          JMP     GNM05
032D    7A      GNM10:  MOV     A,D
032E    FE0D            CPI     CR
0330    C2C802          JNZ     ERROR
0333    01FFFF          LXI     B,0FFFFH
0336    7D              MOV     A,L
0337    B7              ORA     A
0338    CA4003          JZ      GNM20
```

```
033B   C5        GNM15:  PUSH    B
033C   2D                DCR     L
033D   C23B03            JNZ     GNM15
0340   C1        GNM20:  POP     B
0341   D1                POP     D
0342   E1                POP     H
0343   CD5603            CALL    HILO
0346   D24B03            JNC     GNM25
0349   54                MOV     D,H
034A   5D                MOV     E,L


034B   E3        GNM25:  XTHL
034C   D5                PUSH    D
034D   C5                PUSH    B
034E   E5                PUSH    H
034F   3D        GNM30:  DCR     A
0350   F8                RM
0351   E1                POP     H
0352   E3                XTHL
0353   C34F03            JMP     GNM30
                ;*****************************************************************************
                ;       HILO                                                    UTILITY     *
                ;       INPUTS: DE-16 BIT INTEGER                                           *
                ;               HL-16 BIT INTEGER                                           *
                ;       OUTPUTS: CARRY=0 IF HL<DE                                           *
                ;                     =1 IF HL>=DE                                          *
                ;       DESTROYS: A,F/F'S                                                   *
                ;       FUNCTION: HILO COMPARES THE TWO 16-BIT INTEGERS IN HL AND DE PAIRS. *
                ;                 THE INTEGERS ARE TREATED AS UNSIGNED TWO'S COMPLEMENT     *
                ;                 NUMBERS.  THE CARRY BIT IS SET ACCORDING TO THE RESULT    *
                ;                 OF THE COMPARISON.                                        *
                ;*****************************************************************************
0356   C5        HILO:   PUSH    B
0357   47                MOV     B,A
0358   23                INX     H
0359   7C                MOV     A,H
035A   B5                ORA     L
035B   2B                DCX     H
035C   37                STC
035D   CA6503            JZ      HL05
0360   7D                MOV     A,L
0361   93                SUB     E
0362   7C                MOV     A,H
0363   9A                SBB     D
```

```
0364    3F              CMC
0365    78      HL05:   MOV     A,B
0366    C1              POP     B
0367    C9              RET
                ;
0368    3E27    INUST:  MVI     A,CMD
036A    D331            OUT     CNCTL
036C    3ECF            MVI     A,CTCMD
036E    D3F8            OUT     CTCTL
0370    212140          LXI     H,BAUD
0373    7D              MOV     A,L
0374    D3FC            OUT     CTLO
0376    7C              MOV     A,H
0377    D3FD            OUT     CTHI
0379    DB30            IN      CNIN
037B    210238          LXI     H,MSTAK-44
037E    223838          SHLD    SSAVE
0381    312E38          LXI     SP,MSTAK


0384    C38C00          JMP     SOMSG   ;  JUMP TO MONITOR

                ;
                ;*****************************************************************************
                ;       NMOUT                                                   UTILITY     *
                ;       INPUTS: A-8 BIT INTEGER                                             *
                ;       CALLS: ECHO,PRVAL                                                   *
                ;       DESTROYS: A,B,C,F/F'S                                               *
                ;       FUNCTION: NMOUT CONVERTS THE 8 BIT, UNSIGNED INTEGER INT THE A      *
                ;                 REGISTER INTO TWO ASCII CHARACTERS.  THE ASCII CHARACTERS  *
                ;                 ARE THE ONES REPRESENTING THE 8 NITS.  THESE TWO          *
                ;                 CHARACTERS ARE SENT TO THE CONSOLE FOR DISPLAY.           *
                ;*****************************************************************************
0387    F5      NMOUT:  PUSH    PSW
0388    0F              RRC
0389    0F              RRC
038A    0F              RRC
038B    0F              RRC
038C    CD9A03          CALL    PRVAL
038F    CDAF02          CALL    ECHO
0392    F1              POP     PSW
0393    CD9A03          CALL    PRVAL
0396    CDAF02          CALL    ECHO
0399    C9              RET
```

```
                ;*****************************************************************************
                ;       PRVAL                                                   UTILITY     *
                ;       INPUTS: A-INTEGER, RANGE 0 TO F                                     *
                ;       OUTPUTS: A-ASCII CHARACTER                                          *
                ;       DESTROYS: C                                                         *
                ;       FUNCTION: PRVAL CONVERTS THE NUMBER IN THE A REGISTER FROM 0 TO F    *
                ;                 INTO THE CORRESPONDING ASCII CHARACTER, 0-9,A-F.  PRVAL    *
                ;                 DOES NOT CHECK THE VALIDITY OF THE INPUT.                 *
                ;*****************************************************************************
039A    E60F    PRVAL:  ANI     HCHAR
039C    C690            ADI     90H
039E    27              DAA
039F    CE40            ACI     40H
03A1    27              DAA
03A2    4F              MOV     C,A
03A3    C9              RET
                ;
03A4    219D04  REGDS:  LXI     H,RTAB
03A7    4E      REG05:  MOV     C,M
03A8    79              MOV     A,C
03A9    B7              ORA     A
03AA    C2B103          JNZ     REG10
03AD    CDAA02          CALL    CROUT
03B0    C9              RET
03B1    CDAF02  REG10:  CALL    ECHO
03B4    0E3D            MVI     C,'='
03B6    CDAF02          CALL    ECHO


03B9    23              INX     H
03BA    5E              MOV     E,M
03BB    1638            MVI     D,DATA/256
03BD    23              INX     H
03BE    1A              LDAX    D
03BF    CD8703          CALL    NMOUT
03C2    7E              MOV     A,M
03C3    B7              ORA     A
03C4    CACC03          JZ      REG15
03C7    1B              DCX     D
03C8    1A              LDAX    D
03C9    CD8703          CALL    NMOUT
03CC    0E20    REG15:  MVI     C,' '
03CE    CDAF02          CALL    ECHO
03D1    23              INX     H
03D2    C3A703          JMP     REG05
```

```
                        ;
03D5    219D04  RGADR:  LXI     H,RTAB
03D8    110300          LXI     D,RTABS
03DB    7E      RGA05:  MOV     A,M
03DC    B7              ORA     A
03DD    CAC802          JZ      ERROR
03E0    B9              CMP     C
03E1    CAE803          JZ      RGA10
03E4    19              DAD     D
03E5    C3DB03          JMP     RGA05
03E8    23      RGA10:  INX     H
03E9    44              MOV     B,H
03EA    4D              MOV     C,L
03EB    C9              RET
                        ;
03EC    F3      RSTTF:  DI
03ED    312E38          LXI     SP,MSTAK
03F0    D1              POP     D
03F1    C1              POP     B
03F2    F1              POP     PSW
03F3    2A3838          LHLD    SSAVE
03F6    F9              SPHL
03F7    2A3638          LHLD    PSAVE
03FA    E5              PUSH    H
03FB    2A3438          LHLD    LSAVE
03FE    FB              EI
03FF    C9              RET
                        ;*****************************************************************************
                        ;       SRET                                                    UTILITY     *
                        ;       OUTPUTS: CARRY=1                                                    *
                        ;       DESTROYS: CARRY                                                     *
                        ;       FUNCTION: SRET IS CALLED BY ROUTINS WISHING TO RETURN SUCCESS.      *
                        ;                 SRET SETS THE CARRY TRUE AND THEN RETURNS TO THE CALLER   *
                        ;                 OF THE ROUTINE INVOKING SRET.                             *
                        ;*****************************************************************************



0400    37      SRET:   STC
0401    C9              RET
```

```
                        ;
0402    3A3A38  STHF0:  LDA     TEMP
0405    B7              ORA     A
0406    C0              RNZ
0407    0E00            MVI     C,0
0409    CD0D04          CALL    STHLF
040C    C9              RET
                        ;
040D    D5      STHLF:  PUSH    D
040E    E1              POP     H
040F    79              MOV     A,C
0410    E60F            ANI     HCHAR
0412    4F              MOV     C,A
0413    3A3A38          LDA     TEMP
0416    B7              ORA     A
0417    C22004          JNZ     STH05
041A    7E              MOV     A,M
041B    E6F0            ANI     0F0H
041D    B1              ORA     C
041E    77              MOV     M,A
041F    C9              RET
0420    7E      STH05:  MOV     A,M
0421    E60F            ANI     HCHAR
0423    47              MOV     B,A
0424    79              MOV     A,C
0425    0F              RRC
0426    0F              RRC
0427    0F              RRC
0428    0F              RRC
0429    B0              ORA     B
042A    77              MOV     M,A
042B    C9              RET
                ;*****************************************************************************
                ;       VALDG                                                   UTILITY     *
                ;       INPUTS: C-ASCII CHARACTER                                           *
                ;       OUTPUTS: CARRY=1 IF CHARACTER REPRESENTS VALID HEX DIGIT             *
                ;                  =0 OTHERWISE                                             *
                ;       DESTROYS: A,F/F'S                                                   *
                ;       FUNCTION: VALDG RETURNS SUCCESS IF ITS INPUT ARGUMENT IS AN ASCII   *
                ;                 CHARACTER REPRESENTING A VALID HEX DIGIT, AND FAILURE     *
                ;                 OTHERWISE.                                                *
                ;*****************************************************************************
```

```
042C   79         VALDG:  MOV     A,C
042D   FE30                CPI     '0'
042F   FAD302              JM      FRET
0432   FE39                CPI     '9'
0434   FA0004              JM      SRET
0437   CA0004              JZ      SRET
043A   FE41                CPI     'A'

18080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 17

043C   FAD302              JM      FRET
043F   FE47                CPI     'G'
0441   F2D302              JP      FRET
0444   C30004              JMP     SRET
                    ;*****************************************************************************
                    ;       VALDL                                                   UTILITY     *
                    ;       INPUTS: C-CHARACTER                                                 *
                    ;       OUTPUTS: CARRY=1 IF INPUT ARGUMENT IS VALID DELIMETER               *
                    ;                     =0 OTHERWISE                                          *
                    ;       DESTROYS: A,F/F'S                                                   *
                    ;       FUNCTION: VALDL RETURNS SUCCESS IF ITS INPUT ARGUMENT IS A VALID    *
                    ;                 DELIMETER CHARACTER (SPACE, COMMA, CARRIAGE RETURN, EQUAL SIGN)
                    ;                 FAILURE OTHERWISE.                                        *
                    ;*****************************************************************************
0447   79         VALDL:  MOV     A,C
0448   FE2C                CPI     ','
044A   CA0004              JZ      SRET
044D   FE0D                CPI     CR
044F   CA0004              JZ      SRET
0452   FE20                CPI     ' '
0454   CA0004              JZ      SRET
0457   FE3D                CPI     '='
0459   CA0004              JZ      SRET
045C   C3D302              JMP     FRET
                    ;
045F   0D0A4144    SGNON:  DB      CR,LF,'ADSL FM MUSIC SYNTHESIZER',CR,LF
0463   534C2046
0467   4D204D55
046B   53494320
046F   53594E54
0473   48455349
0477   5A45520D
047B   0A
047C   FF                  DB      0FFH
```

```
                        ;
047D  0000      CADR:   DW      0
047F  C401              DW      XCMD
0481  8B01              DW      SCMD
0483  6901              DW      MCMD
0485  EC00              DW      GCMD
0487  C000              DW      DCMD
0489  C004              DW      RCMD
048B  0905              DW      WCMD
048D  0A01              DW      CCMD
048F  4201              DW      ICMD
0491  1301              DW      OCMD
                        ;
0493  4F        CTAB:  DB      'O'
0494  49               DB      'I'
0495  43               DB      'C'
0496  57               DB      'W'


0497  52               DB      'R'
0498  44               DB      'D'
0499  47               DB      'G'
049A  4D               DB      'M'
049B  53               DB      'S'
049C  58               DB      'X'
000A            NCMDS  EQU     $-CTAB
049D  41        RTAB:  DB      'A'
049E  33               DB      ASAVE-((ASAVE/256)*256)
049F  00               DB      0
0003            RTABS  EQU     $-RTAB
04A0  42               DB      'B'
04A1  31               DB      BSAVE-((BSAVE/256)*256)
04A2  00               DB      0
04A3  43               DB      'C'
04A4  30               DB      CSAVE-((CSAVE/256)*256)
04A5  00               DB      0
04A6  44               DB      'D'
04A7  2F               DB      DSAVE-((DSAVE/256)*256)
04A8  00               DB      0
04A9  45               DB      'E'
04AA  2E               DB      ESAVE-((ESAVE/256)*256)
04AB  00               DB      0
04AC  46               DB      'F'
```

```
04AD  32              DB      FSAVE-((FSAVE/256)*256)
04AE  00              DB      0
04AF  48              DB      'H'
04B0  35              DB      HSAVE-((HSAVE/256)*256)
04B1  00              DB      0
04B2  4C              DB      'L'
04B3  34              DB      LSAVE-((LSAVE/256)*256)
04B4  00              DB      0
04B5  4D              DB      'M'
04B6  35              DB      HSAVE-((HSAVE/256)*256)
04B7  01              DB      1
04B8  50              DB      'P'
04B9  37              DB      PSAVE+1-(((PSAVE+1)/256)*256)
04BA  01              DB      1
04BB  53              DB      'S'
04BC  39              DB      SSAVE+1-(((SSAVE+1)/256)*256)
04BD  01              DB      1
04BE  00              DB      0
04BF  00              DB      0
                ;
04C0  CDDD02  RCMD:   CALL    GETHX
04C3  7A              MOV     A,D
04C4  FE0D            CPI     CR
04C6  C2C802          JNZ     ERROR
04C9  C5              PUSH    B
04CA  CD8002  RCM05:  CALL    CI
04CD  4F              MOV     C,A


04CE  CD9302          CALL    CO
04D1  E67F            ANI     PRTYO
04D3  FE3A            CPI     ':'
04D5  C2CA04          JNZ     RCM05
04D8  AF              XRA     A
04D9  57              MOV     D,A
04DA  CD5B05          CALL    BYTE
04DD  5F              MOV     E,A
04DE  CD5B05          CALL    BYTE
04E1  67              MOV     H,A
```

```
04E2  CD5B05          CALL    BYTE
04E5  6F              MOV     L,A
04E6  CD5B05          CALL    BYTE
04E9  B7              ORA     A
04EA  C20305          JNZ     RCM15
04ED  C1              POP     B
04EE  C5              PUSH    B
04EF  09              DAD     B
04F0  4B              MOV     C,E
04F1  CD5B05    RCM10:CALL    BYTE
04F4  77              MOV     M,A
04F5  23              INX     H
04F6  1D              DCR     E
04F7  C2F104          JNZ     RCM10
04FA  CD5B05          CALL    BYTE
04FD  C2C802          JNZ     ERROR
0500  C3CA04          JMP     RCM05
0503  CD5B05    RCM15:CALL    BYTE
0506  C39500          JMP     GETCM
                ;
0509  0E02      WCMD: MVI     C,2
050B  CD1103          CALL    GETNM
050E  D1              POP     D
050F  E1              POP     H
0510  7D        WCM05:MOV     A,L
0511  C610            ADI     10H
0513  4F              MOV     C,A
0514  7C              MOV     A,H
0515  CE00            ACI     0
0517  47              MOV     B,A
0518  7B              MOV     A,E
0519  91              SUB     C
051A  4F              MOV     C,A
051B  7A              MOV     A,D
051C  98              SBB     B
051D  DA2505          JC      WCM10
0520  3E10            MVI     A,10H
0522  C32805          JMP     WCM15
0525  79        WCM10:MOV     A,C
0526  C611            ADI     11H
```

```
0528  B7        WCM15:  ORA     A

0529  CA5505            JZ      WCM25
052C  D5                PUSH    D
052D  5F                MOV     E,A
052E  1600              MVI     D,0
0530  0E3A              MVI     C,':'
0532  CD9302            CALL    CO
0535  7B                MOV     A,E
0536  CD8905            CALL    PBYTE
0539  CD8005            CALL    PADR
053C  AF                XRA     A
053D  CD8905            CALL    PBYTE
0540  7E        WCM20:  MOV     A,M
0541  CD8905            CALL    PBYTE
0544  23                INX     H
0545  1D                DCR     E
0546  C24005            JNZ     WCM20
0549  AF                XRA     A
054A  92                SUB     D
054B  CD8905            CALL    PBYTE
054E  D1                POP     D
054F  CDC105            CALL    PEOL
0552  C31005            JMP     WCM05
0555  CDA605    WCM25:  CALL    PEOF
0558  C3CD02            JMP     EXIT
                ;*****************************************************************************
                ;       BYTE                                                    UTILITY     *
                ;       INPUTS: D-CURRENT VALUE OF CHECKSUM                                 *
                ;       OUTPUTS: A-HEXADECIMAL CHARACTER                                    *
                ;                D-UPDATED VALUE OF CHECKSUM                                *
                ;       CALLS: CI,CNVBN                                                     *
                ;       DESTROYS: A,D,F/F'S                                                 *
                ;       FUNCTION: BYTE READS TWO CHARACTERS FROM THE CONSOLE AND CONVERTS    *
                ;                 THE CHARACTERS TO ONE HEX CHARACTER.  THE A REGISTER      *
                ;                 CONTAINS THE FINAL CHARACTER AND THE D REGISTER CONTAINS   *
                ;                 THE UPDATED VALUE OF THE CHECKSUM.                        *
                ;*****************************************************************************
```

```
055B   C5        BYTE:   PUSH    B
055C   CD8002            CALL    CI
055F   E67F              ANI     PRTYO
0561   4F                MOV     C,A
0562   CD9302            CALL    CO
0565   CD8A02            CALL    CVNBN
0568   07                RLC
0569   07                RLC
056A   07                RLC
056B   07                RLC
056C   47                MOV     B,A
056D   CD8002            CALL    CI
0570   E67F              ANI     PRTYO
0572   4F                MOV     C,A
0573   CD9302            CALL    CO


0576   CD8A02            CALL    CVNBN
0579   B0                ORA     B
057A   4F                MOV     C,A
057B   82                ADD     D
057C   57                MOV     D,A
057D   79                MOV     A,C
057E   C1                POP     B
057F   C9                RET
                  ;*****************************************************************************
                  ;       PADR                                                    UTILITY     *
                  ;       INPUTS: HL-ADDRESS TO BE PRINTED                                    *
                  ;       CALLS: PBYTE                                                        *
                  ;       DESTROYS: A                                                         *
                  ;       FUNCTION: PADR PRINTS THE ADDRESS IN THE HL PAIR BY CALLING PBYTE,  *
                  ;                 THIS CAUSES THE CHECKSUM IN REGISTER D TO BE UPDATED.     *
                  ;*****************************************************************************
0580   7C        PADR:   MOV     A,H
0581   CD8905            CALL    PBYTE
0584   7D                MOV     A,L
0585   CD8905            CALL    PBYTE
0588   C9                RET
```

```
                ;*****************************************************************************
                ;       PBYTE                                                       UTILITY     *
                ;       INPUTS: A-CHARACTER TO BE PRINTED                                       *
                ;               D-CURRENT VALUE OF THE CHECKSUM                                 *
                ;       OUTPUTS: D-UPDATED VALUE OF THE CHECKSUM                                *
                ;       CALLS: PRVAL,CO                                                         *
                ;       DESTROYS: A,F/F'S                                                       *
                ;       FUNCTION: PBYTE CONVERTS THE HEX VALUE IN THE A REGISTER INTO 2         *
                ;                 ASCII CHARACTERS AND OUTPUTS THEM TO THE SERIAL PORT.         *
                ;                 THE CHECKSUM VALUE IN THE D REGISTER IS UPDATED.              *
                ;*****************************************************************************
0589    F5      PBYTE:  PUSH    PSW
058A    0F              RRC
058B    0F              RRC
058C    0F              RRC
058D    0F              RRC
058E    CD9A03          CALL    PRVAL
0591    CD9302          CALL    CO
0594    F1              POP     PSW
0595    F5              PUSH    PSW
0596    CD9A03          CALL    PRVAL
0599    CD9302          CALL    CO
059C    F1              POP     PSW
059D    82              ADD     D
059E    57              MOV     D,A
059F    CD6102          CALL    BREAK
05A2    D0              RNC
05A3    C3CD02          JMP     EXIT
                ;*****************************************************************************
                ;       PEOF                                                        UTILITY     *



                ;       CALLS: CO,PBYTE,PADR                                                    *
                ;       DESTROYS: A,C,D,H,L,F/F'S                                               *
                ;       FUNCTION: PEOF PRINTS THE END OF FILE RECORD CONSISTING OF A RECORD     *
                ;                 MARK, A LOAD ADDRESS OF 0, THE RECORD TYPE, AND THE           *
                ;                 RECORD CHECKSUM.                                              *
                ;*****************************************************************************
```

```
05A6    0E3A        PEOF:   MVI     C,':'
05A8    CD9302              CALL    CO
05AB    AF                  XRA     A
05AC    57                  MOV     D,A
05AD    CD8905              CALL    PBYTE
05B0    210000              LXI     H,0H
05B3    CD8005              CALL    PADR
05B6    3E01                MVI     A,1H
05B8    CD8905              CALL    PBYTE
05BB    AF                  XRA     A
05BC    92                  SUB     D
05BD    CD8905              CALL    PBYTE
05C0    C9                  RET
                    ;
05C1    0E0D        PEOL:   MVI     C,CR
05C3    CD9302              CALL    CO
05C6    0E0A                MVI     C,LF
05C8    CD9302              CALL    CO
05CB    C9                  RET
                    ;
05CC    4F4D4D41    CMSG:   DB      'OMMANDS',CR,LF
05D0    4E44530D
05D4    0A
05D5    432D434F            DB      'C-COMMAND LIST',CR,LF
05D9    4D4D414E
05DD    44204C49
05E1    53540D0A
05E5    442D4449            DB      'D-DISPLAY MEMORY',CR,LF
05E9    53504C41
05ED    59204D45
05F1    4D4F5259
05F5    0D0A
05F7    472D4558            DB      'G-EXECUTE PROGRAM',CR,LF
05FB    45435554
05FF    45205052
0603    4F475241
0607    4D0D0A
060A    492D494E            DB      'I-INPUT FROM PORT',CR,LF
060E    50555420
0612    46524F4D
0616    20504F52
061A    540D0A
```

```
061D  4D2D4D4F          DB      'M-MOVE MEMORY',CR,LF
0621  5645204D
0625  454D4F52



0629  590D0A
062C  4F2D4F55          DB      'O-OUTPUT TO PORT',CR,LF
0630  54505554
0634  20544F20
0638  504F5254
063C  0D0A
063E  522D5245          DB      'R-READ FILE',CR,LF
0642  41442046
0646  494C450D
064A  0A
064B  532D5355          DB      'S-SUBSTITUTE MEMORY',CR,LF
064F  42535449
0653  54555445
0657  204D454D
065B  4F52590D
065F  0A
0660  572D5752          DB      'W-WRITE FILE',CR,LF
0664  49544520
0668  46494C45
066C  0D0A
066E  582D4558          DB      'X-EXAMINE REGISTERS',CR,LF
0672  414D494E
0676  45205245
067A  47495354
067E  4552530D
0682  0A
0683  FF                DB      0FFH
0684  0000              DW      0
                ;
0686  D300      TBL1:   OUT     0
0688  C9                RET
0689  DB00              IN      0
068B  C9                RET
```

```
                        ;
382E                            ORG     REGS
382E                    MSTAK   EQU     $
382E    00              ESAVE:  DB      0
382F    00              DSAVE:  DB      0
3830    00              CSAVE:  DB      0
3831    00              BSAVE:  DB      0
3832    00              FSAVE:  DB      0
3833    00              ASAVE:  DB      0
3834    00              LSAVE:  DB      0
3835    00              HSAVE:  DB      0
3836    0000            PSAVE:  DW      0
3838    0000            SSAVE:  DW      0
383A    00              TEMP:   DB      0
383D                            ORG     BRLOC
0003                    USRBR:  DS      3
38C0                            ORG     RSTORG          ;JUMP TABLE FOR USER RESTARTS AND INTERRUPTS
38C0    C30000          RS1:    JMP     0


38C3    C30000          RS2:    JMP     0
38C6    C30000          RS3:    JMP     0
38C9    C30000          RS4:    JMP     0
38CC    C30000          RS5:    JMP     0
38CF    C30000          RS55:   JMP     0
38D2    C30000          RS6:    JMP     0
38D5    C30000          RS65:   JMP     0
38D8    C30000          RS7:    JMP     0
38DB    C30000          RS75:   JMP     0
38DE    C30000          TRAP:   JMP     0
                                END
NO PROGRAM ERRORS
```

SYMBOL TABLE

* 01

A	0007	ADRD	0247	ADROU	0250	ASAVE	3833
B	0000	BAUD	4021	BR05	0268	BR10	0276
BRCHR	001B	BREAK	0261	BRLOC	383D	BSAVE	3831
BYTE	055B	C	0001	CADR	047D	CCMD	010A
CI	0280	CMD	0027	CMSG	05CC	CNCTL	0031
CNIN	0030	CNOUT	0030	CO	0293	CONST	0031
CR	000D	CROUT	02AA	CSAVE	3830	CTAB	0493
CTCMD	00CF	CTCTL	00F8	CTHI	00FD	CTLO	00FC
CVNBN	028A	D	0002	DATA	3800	DCM05	00C7
DCM10	00CD	DCMD	00C0	DELFA	0010 *	DSAVE	382F
E	0003	ECH05	02B8	ECH10	02C6	ECHO	02AF
ERROR	02C8	ESAVE	382E	ESC	001B	EXIT	02CD
FRET	02D3	FSAVE	3832	GCM05	0101	GCM10	0107
GCMD	00EC	GETCH	02D6	GETCM	0095	GETHX	02DD
GETNM	0311	GHX05	02E3	GHX10	02FB	GNM05	0318
GNM10	032D	GNM15	033B	GNM20	0340	GNM25	034B
GNM30	034F	GO	0076	GTC03	009D *	GTC05	00AA
GTC10	00B6	H	0004	HCHAR	000F	HILO	0356
HL05	0365	HSAVE	3835	ICM01	0153	ICMD	0142
INUST	0368	INVRT	00FF	IOTBL	38F0	KBPRO	0000 *
KBRST	0008 *	KEYPR	0800 *	L	0005	LF	000A
LSAVE	3834	M	0006	MCM05	0177	MCMD	0169
MODE	00CE	MSTAK	382E	NCMDS	000A	NEWLN	0007 *
NMOUT	0387	OC05	0124	OCMD	0113	PADR	0580
PBYTE	0589	PEOF	05A6	PEOL	05C1	PRMSG	029E
PRTYO	007F	PRVAL	039A	PSAVE	3836	PSW	0006
RBR	0002	RCM05	04CA	RCM10	04F1	RCM15	0503
RCMD	04C0	REG05	03A7	REG10	03B1	REG15	03CC
REGDS	03A4	REGS	382E	RGA05	03DB	RGA10	03E8
RGADR	03D5	RIM	0020 *	RS1	38C0	RS2	38C3
RS3	38C6	RS4	38C9	RS5	38CC	RS55	38CF
RS6	38D2	RS65	38D5	RS7	38D8	RS75	38DB
RSTOR	38C0	RSTTF	03EC	RSTU	0038 *	RTAB	049D
RTABS	0003	SCM05	0194	SCM10	019F	SCM15	01AF
SCMD	018B	SETUP	0239	SGNON	045F	SIM	0030

```
SOMSG   008C      SP      0006      SRET    0400      SSAVE   3838
STH05   0420      STHF0   0402 *    STHLF   040D      STLP    0065
STUP    0051      TBL1    0686      TEMP    383A      TERM    001B *
TRAP    38DE      TRDY    0001      TXBE    0004 *    UPPER   00FF *
USRBR   383D *    VALDG   042C      VALDL   0447      WARM    008C
WCM05   0510      WCM10   0525      WCM15   0528      WCM20   0540
WCM25   0555      WCMD    0509      XCM05   01D6      XCM10   01E5
XCM15   01F2      XCM20   0210      XCM25   0227      XCM27   0228
XCM30   0230      XCMD    01C4
```

REFERENCES

[1] James W. Beauchamp, Electronic Music Synthesis class notes (EE302), Fall, 1980.

[2] Jean-Claude Risset and Max V. Mathews, "Analysis of Musical Instrument Tones," PHYSICS TODAY, vol.22, no.2, pp.23-30 (1969).

[3] John M Chowning, "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation," COMPUTER MUSIC JOURNAL, vol.1, April, 1977, pp. 46-54.

[4] Bernard Hutchins, MUSICAL ENGINEER'S HANDBOOK, first edition, Electronotes, 1975.

[5] F. Richard Moore, "Table Lookup Noise for Sinusoidal Digital Oscillators," COMPUTER MUSIC JOURNAL, vol.1, April, 1977, pp.26-29.

[6] John Snell, "Design of a Digital Oscillator Which Will Generate Up to 256 Low Distortion Sine Waves in Real Time," COMPUTER MUSIC JOURNAL, vol.1, April, 1977, p.10.

[7] Texas Instruments Incorporated, THE TTL DATA BOOK, Second Edition, 1976.

[8] Intel Corporation, COMPONENT DATA CATALOG, 1980.

[9] National Semiconductor, LINEAR DATA BOOK, 1978.

[10] Advanced Micro Devices, CONDENSED CATALOG, 1981.

[11] Intel Corporation, MCS-85 USER'S MANUAL, January, 1978.